1. Quantization
    1. Quantization
    2. Memoryless Scalar Quantization
    3. MSE-Optimal Memoryless Scalar Quantization
    4. Entropy Coding
    5. Quantizer Design for Entropy Coded Sytems
    6. Adaptive Quantization
2. DPCM
    1. Pulse Code Modulation
    2. Differential Pulse Code Modulation
    3. Performance of DPCM
    4. Analysis of DPCM using Rate-Distortion Theory
3. Transform Coding
    1. Transform Coding: Background and Motivation
    2. Optimal Bit Allocation
    3. Gain over PCM
    4. The Optimal Orthogonal Transform
    5. Performance
    6. Sub-Optimum Orthogonal Transforms
4. Subband Coding
    1. Introduction and Motivation
    2. Fundamentals of Multirate Signal Processing
    3. Uniformly-Modulated Filterbanks
    4. MPEG Layers 1-3: Cosine-Modulated Filterbanks
    5. MP3 and AAC: MDCT Processing

Quantization

This module introduces the topic of quantization and prefaces the discussion of later modules in the course on source coding.

- Given a continuous-time and continuous-amplitude signal $x(t)$, processing and storage by modern digital hardware requires discretization in both time and amplitude, as accomplished by an analog-to-digital converter (ADC).
- We will typically work with discrete-time quantities $x(n)$ (indexed by "time" variable *n*) which we assume were sampled ideally and without aliasing.
- Here we discuss various ways of discretizing the amplitude of $x(n)$ so that it may be represented by a finite set of numbers. This is generally a lossy operation, and so we analyze the performance of various quantization schemes and design quantizers that are optimal under certain assumptions. A good reference for much of this material is the textbook by Jayant and Noll.

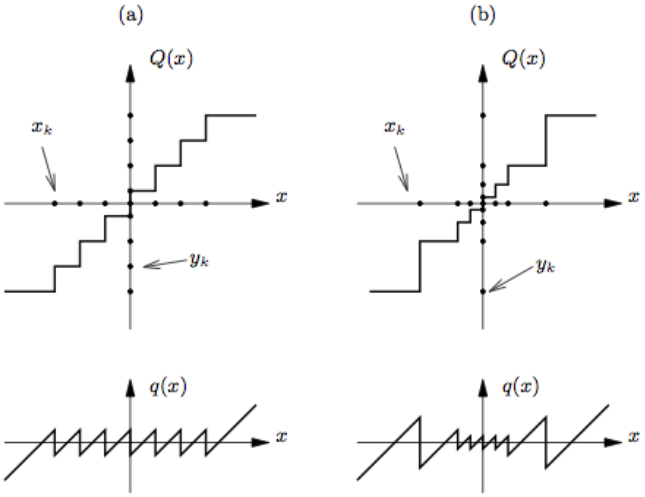
Memoryless Scalar Quantization

Memoryless scalar quantization is discussed, with a focus on the uniform quantizer. Uniform quantizer error variance is derived under the assumption of many quantization levels, and several examples are provided.

- *Memoryless scalar quantization* of continuous-amplitude variable *x* is the mapping of *x* to output y_k when *x* lies within interval
  Equation:

$$\mathscr{X}_k := \{x_k < x \leq x_{k+1}\},\ k = 1, 2, \cdots, L.$$

  The x_k are called *decision thresholds*, and the number of quantization levels is *L*. The quantization operation is written $y = Q(x)$.
- When $0 \in \{y_1, \cdots, y_L\}$, quantizer is called *midtread*, else *midrise*.
- *Quantization error* defined $q := x - Q(x)$



(a) Uniform and (b) non-uniform quantization Q(x) and quantization error q(x)

- If *x* is a r.v. with pdf $p_x(\cdot)$ and likewise for *q*, then quantization error variance is
  Equation:

$$\begin{aligned}\sigma_q^2 = \mathrm{E}\left\{q^2\right\} &= \int_{-\infty}^{\infty} q^2 p_q(q)dq \\ &= \int_{-\infty}^{\infty} (x - Q(x))^2 p_x(x)dx \\ &= \sum_{k=1}^{L} \int_{x_k}^{x_{k+1}} (x - y_k)^2 p_x(x)dx\end{aligned}$$

- A special quantizer is the *uniform quantizer*:
  Equation:

$$\begin{aligned} y_{k+1} - y_k &= \Delta, \quad \text{for } k = 1, 2, \cdots, L-1, \\ x_{k+1} - x_k &= \Delta, \quad \text{for finite } x_k, x_{k+1}, \\ -x_1 = x_{L+1} &= \infty. \end{aligned}$$

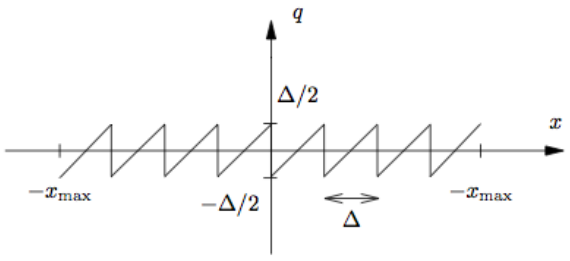
- *Uniform Quantizer Performance for large L:* For bounded input $x \in (-x_{\max}, x_{\max})$, uniform quantization with $x_2 = -x_{\max} + \Delta$ and $x_L = x_{\max} - \Delta$, and with $y_1 = x_2 - \Delta/2$ and $y_k = x_k + \Delta/2$ (for $k > 1$), the quantization error is well approximated by a uniform distribution for large L:
  Equation:

$$p_q(q) = \begin{cases} 1/\Delta & |q| \le \Delta/2, \\ 0 & \text{else.} \end{cases}$$

  Why?

  - As $L \to \infty$, $p_x(x)$ is constant over X_k for any *k*. Since $q = x - y_k|_{x \in \mathscr{X}_k}$, it follows that $p_q(q|x \in \mathscr{X}_k)$ will have uniform distribution for any *k*.
  - With $x \in (-x_{\max}, x_{\max})$ and with x_k and y_k as specified, $q \in (-\Delta/2, \Delta/2]$ for all *x* (see [link]). Hence, for any *k*,
    Equation:

$$p_q(q|x \in \mathscr{X}_k) = \begin{cases} 1/\Delta & q \in (-\Delta/2, \Delta/2], \\ 0 & \text{else.} \end{cases}$$



Quantization error for bounded input and midpoint y_k

In this case, from [link] (upper equation),
Equation:

$$\sigma_q^2 = \int_{-\Delta/2}^{\Delta/2} q^2 \frac{1}{\Delta}\, dq = \frac{1}{\Delta}\left[\frac{q^3}{3}\right]_{-\Delta/2}^{\Delta/2} = \frac{1}{\Delta}\left(\frac{\Delta^3}{3 \cdot 8} + \frac{\Delta^3}{3 \cdot 8}\right) = \boxed{\frac{\Delta^2}{12}}.$$

If we use *R* bits to represent each discrete output *y* and choose $L = 2^R$, then
Equation:

$$\sigma_q^2 = \frac{\Delta^2}{12} = \frac{1}{12}\left(\frac{2x_{\max}}{L}\right)^2 = \frac{1}{3}x_{\max}^2\, 2^{-2R}$$

and
Equation:

$$\text{SNR } [\text{dB}] \;=\; 10\log_{10}\left(\frac{\sigma_x^2}{\sigma_q^2}\right) \;=\; 10\log_{10}\left(3\frac{\sigma_x^2}{x_{\max}^2}2^{2R}\right) \;=\; \boxed{6.02R - 10\log_{10}\left(3\frac{x_{\max}^2}{\sigma_x^2}\right)}.$$

Recall that the expression above is only valid for σ_x^2 small enough to ensure $x \in (-x_{\max}, x_{\max})$. For larger σ_x^2, the quantizer *overloads* and the SNR decreases rapidly.

Example:
SNR for Uniform Quantization of Uniformly-Distributed Input
For uniformly distributed *x*, can show $x_{\max}/\sigma_x = \sqrt{3}$, so that $\text{SNR} = 6.02R$.

Example:
SNR for Uniform Quantization of Sinusoidal Input)
For a sinusoidal *x*, can show $x_{\max}/\sigma_x = \sqrt{2}$, so that $\text{SNR} = 6.02R + 1.76$. (Interesting since sine waves are often used as test signals).

Example:
SNR for Uniform Quantization of Gaussian Input
Though not truly bounded, Gaussian *x* might be considered as approximately bounded if we choose $x_{\max} = 4\sigma_x$ and ignore residual clipping. In this case $\text{SNR} = 6.02R - 7.27$.

MSE-Optimal Memoryless Scalar Quantization

The mean-squared error minimizing scalar quantizer (the Lloyd-Max quantizer) is derived here using Lagrange optimization. Background on Lagrange optimization is also provided. Finally, error variance is derived for the asymptotic case of many quantization levels.

- Though uniform quantization is convenient for implementation and analysis, non-uniform quantization yeilds lower $\sigma_q{}^2$ when $p_x(\bullet)$ is non-uniformly distributed. By decreasing $|q(x)|$ for frequently occuring x (at the expense of increasing $|q(x)|$ for infrequently occuring x), the average error power can be reduced.
- *Lloyd-Max Quantizer:* MSE-optimal thresholds $\{x_k\}$ and outputs $\{y_k\}$ can be determined given an input distribution $p_x(\bullet)$, and the result is the *Lloyd-Max quantizer*. Necessary conditions on $\{x_k\}$ and $\{y_k\}$ are
  Equation:

$$\frac{\partial \sigma_q^2}{\partial x_k} = 0 \quad \text{for} \quad k \in \{2, \cdots, L\} \quad \text{and} \quad \frac{\partial \sigma_q^2}{\partial y_k} = 0 \quad \text{for} \quad k \in \{1, \cdots, L\}.$$

  Using equation 2 from Memoryless Scalar Quantization (third equation), $\partial/\partial b \int_a^b f(x)dx = f(b)$, $\partial/\partial a \int_a^b f(x)dx = -f(a)$, and above,
  Equation:

$$\frac{\partial \sigma_q^2}{\partial x_k} = (x_k - y_{k-1})^2 p_x(x_k) - (x_k - y_k)^2 p_x(x_k) = 0 \quad \Rightarrow \quad \boxed{\begin{array}{l} x_k^\star = \frac{y_k^\star + y_{k-1}^\star}{2}, k \in \{2 \cdots L\}, \\ x_1^\star = -\infty, x_{L+1}^\star = \infty, \end{array}}$$

$$\frac{\partial \sigma_q^2}{\partial y_k} = 2 \int_{x_k}^{x_{k+1}} (x - y_k) p_x(x)dx = 0 \quad \Rightarrow \quad \boxed{y_k^\star = \frac{\int_{x_k^\star}^{x_{k+1}^\star} x p_x(x)dx}{\int_{x_k^\star}^{x_{k+1}^\star} p_x(x)dx}, k \in \{1 \cdots L\}}$$

  It can be shown that above are sufficient for global MMSE when $\partial^2 \log p_x(x)/\partial x^2 \leq 0$, which holds for uniform, Gaussian, and Laplace pdfs, but not Gamma. *Note:*

  - *optimum decision thresholds are halfway between neighboring output values,*
  - *optimum output values are centroids of the pdf within the appropriate interval, i.e., are given by the conditional means*
    Equation:

$$y_k^\star = \mathrm{E}\{x | x \in \mathscr{X}_k^\star\} = \int x p_x(x | x \in \mathscr{X}_k^\star)dx = \int x \frac{p_x(x, x \in \mathscr{X}_k^\star)}{\Pr\left(x \in \mathscr{X}_k^\star\right)} dx, = \frac{\int_{x_k^\star}^{x_{k+1}^\star} x p_x(x)dx}{\int_{x_k^\star}^{x_{k+1}^\star} p_x(x)dx}.$$

  Iterative Procedure to Find $\{x_k^\star\}$ and $\{y_k^\star\}$:

  1. *Choose $\hat{y}_1$.*
  2. *For $k = 1, \cdots, L-1$,*
     given $\hat{y}_k$ and $\widehat{x}_k$, solve [link] (lower equation) for $\widehat{x}_{k+1}$,
     given $\hat{y}_k$ and $\widehat{x}_{k+1}$, solve [link] (upper equation) for $\hat{y}_{k+1}$.end;
  3. *Compare $\hat{y}_L$ to y_L calculated from [link] (lower equation) based on $\widehat{x}_L$ and $x_{L+1} = \infty$. Adjust $\hat{y}_1$ accordingly, and go to step 1.*

- *Lloyd-Max Performance for large L:* *As with the uniform quantizer, can analyze quantization error performance for large L. Here, we assume that*
  - *the pdf $p_x(x)$ is constant over $x \in \mathscr{X}_k$ for $k \in \{1, \cdots, L\}$,*
  - *the pdf $p_x(x)$ is symmetric about $x = 0$,*
  - *the input is bounded, i.e., $x \in (-x_{\max}, x_{\max})$ for some (potentially large) x_{max}.*

  So with assumption
  Equation:

$$p_x(x) = p_x(y_k) \quad \text{for} \quad x, y_k \in \mathscr{X}_k$$

  and definition
  Equation:

$$\Delta_k \;:=\; x_{k+1} - x_k,$$

  we can write
  Equation:

$$P_k \;:= \Pr\{x \in \mathscr{X}_k\} \;=\; p_x(y_k)\Delta_k \quad \left(\text{where we require} \sum P_k = 1\right)$$

  and thus, from equation 2 from Memoryless Scalar Quantization(lower equation), σ_q^2 becomes
  Equation:

$$\sigma_q^2 \;=\; \sum_{k=1}^{L} \frac{P_k}{\Delta_k} \int_{x_k}^{x_{k+1}} (x - y_k)^2 dx.$$

  For MSE-optimal $\{y_k\}$, know
  Equation:

$$0 \;=\; \frac{\partial \sigma_q^2}{\partial y_k} \;=\; 2\frac{P_k}{\Delta_k} \int_{x_k}^{x_{k+1}} (x - y_k) dx \quad \Rightarrow \quad \boxed{y_k^\star = \tfrac{x_k + x_{k+1}}{2}},$$

  which is expected since the centroid of a flat pdf over X_k is simply the midpoint of X_k. Plugging $y_k^\star$ into [link],
  Equation:

$$\begin{aligned}
\sigma_q^2 \;&=\; \sum_{k=1}^{L} \frac{P_k}{3\Delta_k} \Big[(x - x_k/2 - x_{k+1}/2)^3\Big]_{x_k}^{x_{k+1}} \\
&=\; \sum_{k=1}^{L} \frac{P_k}{3\Delta_k} \Big[(x_{k+1}/2 - x_k/2)^3 - (x_k/2 - x_{k+1}/2)^3\Big] \\
&=\; \sum_{k=1}^{L} \frac{P_k}{3\Delta_k} \bullet 2\left(\frac{\Delta_k}{2}\right)^3 \;=\; \frac{1}{12} \sum_{k=1}^{L} P_k \Delta_k^2.
\end{aligned}$$

  Note that for uniform quantization ($\Delta_k = \Delta$), the expression above reduces to the one derived earlier. Now we minimize σ_q^2 w.r.t. $\{\Delta_k\}$. The trick here is to define
  Equation:

$$\alpha_k \;:=\; \sqrt[3]{p_x\left(y_k^\star\right)}\Delta_k \quad \text{so that} \quad \sigma_q^2 \;=\; \frac{1}{12}\sum_{k=1}^{L} p_x\left(y_k^\star\right)\Delta_k^3 \;=\; \frac{1}{12}\sum_{k=1}^{L}\alpha_k^3.$$

For $p_x(x)$ constant over X_k and $y_k \in \mathscr{X}_k$,
Equation:

$$\sum_{k=1}^{L}\alpha_k \;=\; \sum_{k=1}^{L}\sqrt[3]{p_x\left(y_k^\star\right)}\Delta_k\Big|_{y_k^\star=\frac{x_k+x_{k+1}}{2}} \;=\; \int_{-x_{\max}}^{x_{\max}}\sqrt[3]{p_x(x)}dx \;=\; C_x \;\;(\text{a known constant}),$$

we have the following constrained optimization problem:
Equation:

$$\min_{\{\alpha_k\}}\sum_k \alpha_k^3 \quad \text{s.t.} \quad \sum_k \alpha_k = C_x.$$

This may be solved using Lagrange multipliers.

Note:
Optimization via Lagrange Multipliers
Consider the problem of minimizing N-dimensional real-valued cost function $J(\mathbf{x})$, where $\mathbf{x}=(x_1,x_2,\cdots,x_N)^t$, subject to $M<N$ real-valued equality constraints $f_m(\mathbf{x})=a_m$, $m=1,\cdots,M$. This may be converted into an unconstrained optimization of dimension $N+M$ by introducing additional variables $\lambda=(\lambda_1,\cdots,\lambda_M)^t$ known as Lagrange multipliers. The uncontrained cost function is
Equation:

$$J_u(\mathbf{x},\lambda) \;=\; J(\mathbf{x})+\sum_m \lambda_m(f_m(\mathbf{x})-a_m),$$

and necessary conditions for its minimization are
Equation:

$$\frac{\partial}{\partial \mathbf{x}}J_u(\mathbf{x},\lambda)=\mathbf{0} \;\Leftrightarrow\; \frac{\partial}{\partial \mathbf{x}}J(\mathbf{x}) \;+\sum_m \lambda_m\frac{\partial}{\partial \mathbf{x}}f_m(\mathbf{x})=\mathbf{0}$$
$$\frac{\partial}{\partial \lambda}J_u(\mathbf{x},\lambda)=\mathbf{0} \;\Leftrightarrow\; f_m(\mathbf{x})=a_m \quad \text{for} \quad m=1,\cdots,M.$$

*The typical procedure used to solve for optimal **x** is the following:*

1. *Equations for x_n, $n=1,\cdots,N$, in terms of $\{\lambda_m\}$ are obtained from [link] (upper equation).*
2. *These N equations are used in [link] (lower equation) to solve for the M optimal λ_m.*
3. *The optimal $\{\lambda_m\}$ are plugged back into the N equations for x_n, yielding optimal $\{x_n\}$.*

Necessary conditions are
Equation:

$$\forall \ell,\ \frac{\partial}{\partial \alpha_\ell}\left(\sum_k \alpha_k^3 + \lambda\left(\sum_k \alpha_k - C_x\right)\right) = 0 \ \Rightarrow\ \lambda = -3\alpha_\ell^2 \ \Rightarrow\ \alpha_\ell = \sqrt{-\lambda/3}$$

$$\frac{\partial}{\partial \lambda}\left(\sum_k \alpha_k^3 + \lambda\left(\sum_k \alpha_k - C_x\right)\right) = 0 \ \Rightarrow\ \sum_k \alpha_k = C_x,$$

which can be combined to solve for λ:
Equation:

$$\sum_{k=1}^{L} \sqrt{-\frac{\lambda}{3}} = C_x \ \Rightarrow\ \lambda = -3\left(\frac{C_x}{L}\right)^2.$$

Plugging λ back into the expression for α_ℓ, we find
Equation:

$$\alpha_\ell^\star \ = \ C_x/L, \quad \forall\, \ell.$$

Using the definition of α_ℓ, the optimal decision spacing is
Equation:

$$\Delta_k^\star \ = \ \frac{C_x}{L\sqrt[3]{p_x\left(y_k^\star\right)}} \ = \ \boxed{\frac{\int_{-x_{\max}}^{x_{\max}} \sqrt[3]{p_x(x)}dx}{L\sqrt[3]{p_x(y_k^\star)}}},$$

and the minimum quantization error variance is
Equation:

$$\begin{aligned}\sigma_q^2\,\|\min \ &= \ \frac{1}{12}\sum_k p_x\left(y_k^\star\right)\Delta_k^{\star 3} \ = \ \frac{1}{12}\sum_k p_x\left(y_k^\star\right)\frac{\left(\int_{-x_{\max}}^{x_{\max}} \sqrt[3]{p_x\left(x\right)}dx\right)^3}{L^3 p_x\left(y_k^\star\right)} \\ &= \ \boxed{\frac{1}{12L^2}\left(\int_{-x_{\max}}^{x_{\max}} \sqrt[3]{p_x\left(x\right)}dx\right)^3}.\end{aligned}$$

An interesting observation is that $\alpha_\ell^{\star 3}$, the ℓ^{th} interval's optimal contribution to ${\sigma_q}^2$, is constant over ℓ.

Entropy Coding

In this module, the concepts of entropy and variable-length coding are introduced, motivating the description of the Huffman encoder.

- *Binary Scalar Encoding:* Previously we have focused on the memoryless scalar quantizer $y = Q(x)$, where *y* takes a value from a set of *L* reconstruction levels. By coding each quantizer output in binary format, we transmit (store) the information at a rate (cost) of
  Equation:

$$R = \lceil \log_2 L \rceil \quad \text{bits/sample.}$$

  If, for example, $L = 8$, then we transmit at 3 bits/sample. Say we can tolerate a bit more quantization error, e.g., as results from $L = 5$. We hope that this reduction in fidelity reduces our transmission requirements, but with this simple binary encoding scheme we still require $R = 3$ bits/sample!
- *Idea—Block Coding:* Let's assign a symbol to each block of 3 consecutive quantizer outputs. We need a symbol alphabet of size $\geq 5^3 = 125$, which is adequately represented by a 7-bit word ($2^7 = 128$). Transmitting these words requires only $7/3 = 2.33$ bits/sample!
- *Idea—Variable Length Coding:* Assume some of the quantizer outputs occur more frequently than others. Could we come up with an alphabet consisting of short words for representing frequent outputs and longer words for infrequent outputs that would have a lower *average* transmission rate?

Example:
Variable Length Coding)
Consider the quantizer with $L = 4$ and output probabilities indicated in [link]. Straightforward 2-bit encoding requires average bit rate of 2 bits/sample, while the variable length code in [link] gives average
$R = \sum_k P_k n_k = 0.6 \cdot 1 + 0.25 \cdot 2 + 0.1 \cdot 3 + 0.05 \cdot 3 = 1.55$ bits/sample.

output	P_k	code
y_1	0.60	0
y_2	0.25	01

y_3	0.10	011
y_4	0.05	111

- *(Just enough information about) Entropy:*

Exercise:

Problem:

Given an arbitrarily complex coding scheme, what is the minimum bits/sample required to transmit (store) the sequence $\{y(n)\}$?

Solution:

When random process $\{y(n)\}$ is i.i.d., the minimum average bit rate is

Equation:

$$R_{\min} \;=\; H_y + \epsilon,$$

where H_y is the *entropy* of random variable $y(n)$ in bits:

Equation:

$$H_y \;=\; -\sum_{k=1}^{L} P_k \log_2 P_k,$$

and ϵ is an arbitrarily small positive constant (see textbooks by Berger and by Cover & Thomas).

Notes:

- Entropy obeys $0 \leq H_y \leq \log_2 L$. The left inequality occurs when $P_k = 1$ for some *k*, while the right inequality occurs when $P_k = 1/L$ for every *k*.
- The term *entropy* refers to the average information of a single random variable, while the term *entropy rate* refers to a sequence of random variables, i.e., a random process.
- When $\{y(n)\}$ is not independent (the focus of later sections), a different expression for R_{min} applies.
- Though the minimum rate is well specified, the construction of a coding scheme which always achieves this rate is not.

Example:
Entropy of Variable Length Code
Recalling the setup of [link], we find that
$H_y = -(0.6 \log_2 0.6 + 0.25 \log_2 0.25 + 0.1 \log_2 0.1 + 0.05 \log_2 0.05) = 1.49$ bits. Assuming i.i.d. $\{y(n)\}$, we have $R_{\min} = 1.49$ bits/sample. Compare to the variable length code on the right which gave $R = 1.55$ bits/sample.

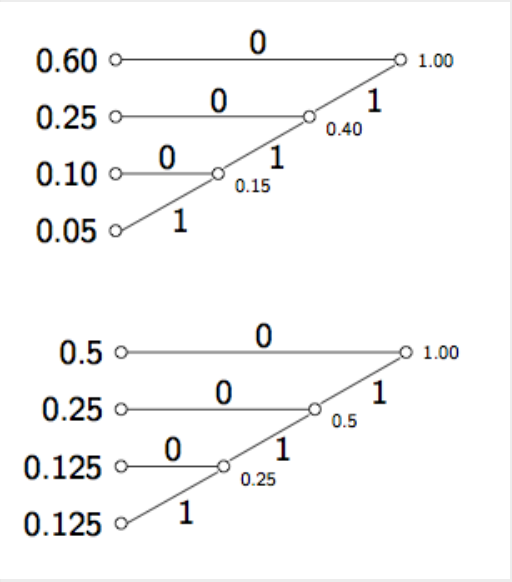
output	P_k	code
y_1	0.60	0
y_2	0.25	01
y_3	0.10	011
y_4	0.05	111

- *Huffman Encoding:* Given quantizer outputs y_k or fixed-length blocks of outputs $(y_j\, y_k\, y_\ell)$, the *Huffman* procedure constructs variable length codes that are optimal in certain respects (see Cover & Thomas). For example, when the probabilities of $\{P_k\}$ are powers of 1/2 (and $\{y(n)\}$ is i.i.d.), the entropy rate of a Huffman encoded output attains R_{min}.

Note:
Huffman Procedure (Binary Case)

1. Arrange ouput probabilities P_k in decreasing order and consider them as leaf nodes of a tree.
2. While there exists more than one node:
    - Merge the two nodes with smallest probability to form a new node whose probability equals the sum of the two merged nodes.
    - Arbitrarily assign 1 and 0 to the two branches of the merging pair.

3. The code assigned to each output is obtained by reading the branch bits sequentially from root note to leaf node.



Example:
Huffman Encoder Attaining R_{min}
In [link], a Huffman code was constructed for the output probabilities listed below. Here $H_y = -(0.5 \log_2 0.5 + 0.25 \log_2 0.25 + 2 \cdot 0.125 \log_2 0.125) = 1.75$ bits, so that $R_{\min} = 1.75$ bits/sample (with the i.i.d. assumption). Since the average bit rate for the Huffman code is also $R = 0.5 \cdot 1 + 0.25 \cdot 2 + 0.125 \cdot 3 + 0.125 \cdot 3 = 1.75$ bits/sample, Huffman encoding attains R_{min} for this output distribution.

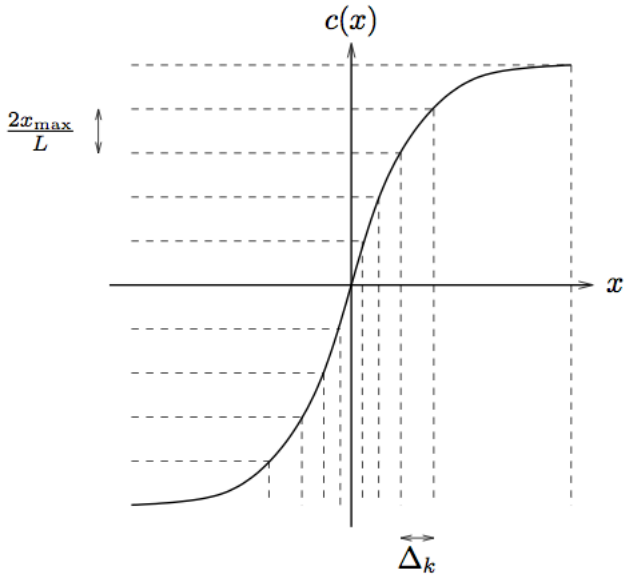
output	P_k	code
y_1	0.5	0
y_2	0.25	01
y_3	0.125	011
y_4	0.125	111

Quantizer Design for Entropy Coded Sytems

Motivated by the cascade of memoryless quantization and entropy coding, the entropy-minimizing scalar memoryless quantizer is derived. Using a compander formulation and tools from the calculus of variations, it is shown that the entropy-minimizing quantizer is the simple uniform quantizer. The penalty associated with memoryless quantization is then analyzed in the asymptotic case of many quantization levels.

- Say that we are designing a system with a memoryless quantizer followed by an entropy coder, and our goal is to minimize the average transmission rate for a given σ_q^2 (or vice versa). Is it optimal to cascade a σ_q^2-minimizing (Lloyd-Max) quantizer with a rate-minimizing code? In other words, what is the optimal memoryless quantizer if the quantized outputs are to be entropy coded?
- *A Compander Formulation:* To determine the optimal quantizer,

  1. consider a *companding* system: a memoryless nonlinearity $c(x)$ followed by uniform quantizer,
  2. find $c(x)$ minimizing entropy H_y for a fixed error variance σ_q^2.



Compander curve: nonuniform input regions mapped to uniform output regions (for subsequent uniform quantization)

- First we must express σ_q^2 and H_y in terms of $c(x)$. [link] suggests that, for large L, the slope $c'(x) := dc(x)/dx$ obeys

  Equation:

$$c'(x)|_{x\in\mathcal{X}_k} = \frac{2x_{\max}/L}{\Delta_k},$$

  so that we may write

  Equation:

$$\Delta_k = \frac{2x_{\max}}{Lc'(x)}_{x\in\mathcal{X}_k}.$$

  Assuming large L, the σ_q^2-approximation equation 9 from MSE-Optimal Memoryless Scalar Quantization (lower equation) can be transformed as follows.

  Equation:

$$\begin{aligned}
\sigma_q^2 &= \frac{1}{12}\sum_{k=1}^{L} P_k \Delta_k^2 \\
&= \frac{x_{\max}^2}{3L^2}\sum_{k=1}^{L}\frac{P_k}{c'(x)^2}\quad{}_{x\in\mathscr{X}_k} \\
&= \frac{x_{\max}^2}{3L^2}\sum_{k=1}^{L}\frac{p_x(x)}{c'(x)^2}\Delta_k \quad{}_{x\in\mathscr{X}_k}\ \text{since} P_k = p_x(x)\quad{}_{x\in\mathscr{X}_k}\ \Delta_k \quad \text{for large } L \\
&= \frac{x_{\max}^2}{3L^2}\int_{-x_{\max}}^{x_{\max}}\frac{p_x(x)}{c'(x)^2}dx.
\end{aligned}$$

Similarly,
Equation:

$$\begin{aligned}
H_y &= -\sum_{k=1}^{L} P_k \log_2 P_k \\
&= -\sum_{k=1}^{L} p_x(x)\Delta_k \log_2\left(p_x(x)\Delta_k\right)\quad{}_{x\in\mathscr{X}_k} \\
&= -\sum_{k=1}^{L} p_x(x)\Delta_k \log_2 p_x(x)\quad{}_{x\in\mathscr{X}_k} - \sum_{k=1}^{L} p_x(x)\Delta_k \log_2 \Delta_k\quad{}_{x\in\mathscr{X}_k} \\
&= -\int_{-x_{\max}}^{x_{\max}} p_x(x)\log_2 p_x(x)dx - \int_{-x_{\max}}^{x_{\max}} p_x(x)\log_2\frac{2x_{\max}}{Lc'(x)}dx \\
&\qquad {}_{h_x:\text{``differential entropy''}} \qquad\qquad\qquad\qquad\qquad\qquad {}_{\Delta_k} \\
&= h_x - \log_2\frac{2x_{\max}}{L}\underbrace{\int_{-x_{\max}}^{x_{\max}} p_x(x)dx}_{=1} + \int_{-x_{\max}}^{x_{\max}} p_x(x)\log_2 c'(x)dx \\
&= \text{constant} + \int_{-x_{\max}}^{x_{\max}} p_x(x)\log_2 c'(x)dx
\end{aligned}$$

- *Entropy-Minimizing Quantizer:* Our goal is to choose $c(x)$ which minimizes the entropy rate H_y subject to fixed error variance ${\sigma_q}^2$. We employ a Lagrange technique again, minimizing the cost $\int_{-x_{\max}}^{x_{\max}} p_x(x)\log_2 c'(x)dx$ under the constraint that the quantity $\int_{-x_{\max}}^{x_{\max}} p_x(x)(c'(x))^{-2}dx$ equals a constant C. This yields the unconstrained cost function
Equation:

$$J_u\left(c'(x),\lambda\right) = \int_{-x_{\max}}^{x_{\max}}\Big[p_x(x)\log_2 c'(x) + \lambda\underbrace{\Big(p_x(x)\big(c'(x)\big)^{-2} - C\Big)}_{\varphi(c'(x),\lambda)}\Big]dx,$$

with scalar λ, and the unconstrained optimization problem becomes
Equation:

$$\min_{c'(x),\lambda} J_u\left(c'(x),\lambda\right).$$

The following technique is common in variational calculus (see, e.g., Optimal Systems Control by Sage & White). Say $a^\star(x)$ minimizes a (scalar) cost $J(a(x))$. Then for *any* (well-behaved) variation $\eta(x)$ from this optimal $a^\star(x)$, we must have
Equation:

$$\frac{\partial}{\partial \epsilon} J(a^{\star}(x)+\epsilon\eta(x))\Big|_{\epsilon=0}=0$$

where ϵ is a scalar. Applying this principle to our optimization problem, we search for $c'(x)$ such that
Equation:

$$\forall \eta(x),\quad \frac{\partial}{\partial \epsilon} J_u\big(c'(x)+\epsilon\eta(x),\lambda\big)\Big|_{\epsilon=0}=0.$$

From [link] we find (using $\log_2 a = \log_2 e \cdot \log_e a$)
Equation:

$$\begin{aligned}
\frac{\partial J_u}{\partial \epsilon}\Big|_{\epsilon=0} &= \int_{-x_{\max}}^{x_{\max}} \frac{\partial}{\partial \epsilon}\varphi\big(c'(x)+\epsilon\eta(x),\lambda\big)\Big|_{\epsilon=0} dx \\
&= \int_{-x_{\max}}^{x_{\max}} \frac{\partial}{\partial \epsilon}\Big[p_x(x)\log_2(e)\,\log_e\big(c'(x)+\epsilon\eta(x)\big)+\lambda\Big(p_x(x)\big(c'(x)+\epsilon\eta(x)\big)^{-2}-C\Big)\Big]_{\epsilon=} \\
&= \int_{-x_{\max}}^{x_{\max}} \Big[\log_2(e)\,p_x(x)\big(c'(x)+\epsilon\eta(x)\big)^{-1}\eta(x)-2\lambda p_x(x)\big(c'(x)+\epsilon\eta(x)\big)^{-3}\eta(x)\Big]_{\epsilon=0} dx \\
&= \int_{-x_{\max}}^{x_{\max}} p_x(x)\big(c'(x)\big)^{-1}\Big[\log_2(e)-2\lambda\big(c'(x)\big)^{-2}\Big]\eta(x)\,dx
\end{aligned}$$

and to allow for any $\eta(x)$ we require
Equation:

$$\log_2(e)-2\lambda\big(c'(x)\big)^{-2}=0 \quad\Leftrightarrow\quad c'(x)=\underbrace{\sqrt{\frac{2\lambda}{\log_2 e}}}_{\text{a constant!}}.$$

Applying the boundary conditions,
Equation:

$$\begin{aligned} c(x_{\max}) &= x_{\max} \\ c(-x_{\max}) &= -x_{\max} \end{aligned} \quad\rightarrow\quad \boxed{c(x)=x}.$$

Thus, for large-L, *the quantizer that minimizes entropy rate H_y for a given quantization error variance σ_q^2 is the uniform quantizer.* Plugging $c(x)=x$ into [link], the rightmost integral disappears and we have
Equation:

$$H_y|_{\text{uniform}} = h_x - \log_2 \underbrace{\frac{2x_{\max}}{L}}_{\Delta},$$

and using the large-L uniform quantizer error variance approximation equation 6 from Memoryless Scalar Quantization,
Equation:

$$H_y|_{\text{uniform}} = h_x - \frac{1}{2}\log_2\big(12\sigma_q^2\big).$$

It is interesting to compare this result to the information-theoretic minimal average rate for transmission of a continuous-amplitude memoryless source x of differential entropy h_x at average distortion σ_q^2(see Jayant & Noll or Berger):
Equation:

$$R_{\min} \;=\; h_x - \frac{1}{2}\,\log_2\left(2\pi e\,\sigma_q^2\right).$$

Comparing the previous two equations, we find that (for a continous-amplitude memoryless source) uniform quantization prior to entropy coding requires
Equation:

$$\frac{1}{2}\,\log_2\left(\frac{\pi e}{6}\right) \;\approx\; \boxed{0.255\ \ \text{bits/sample}}$$

more than the theoretically optimum transmission scheme, regardless of the distribution of *x*. Thus, *0.255 bits/sample (or* ~ 1.5 *dB using the* $6.02R$ *relationship) is the price paid for memoryless quantization*.

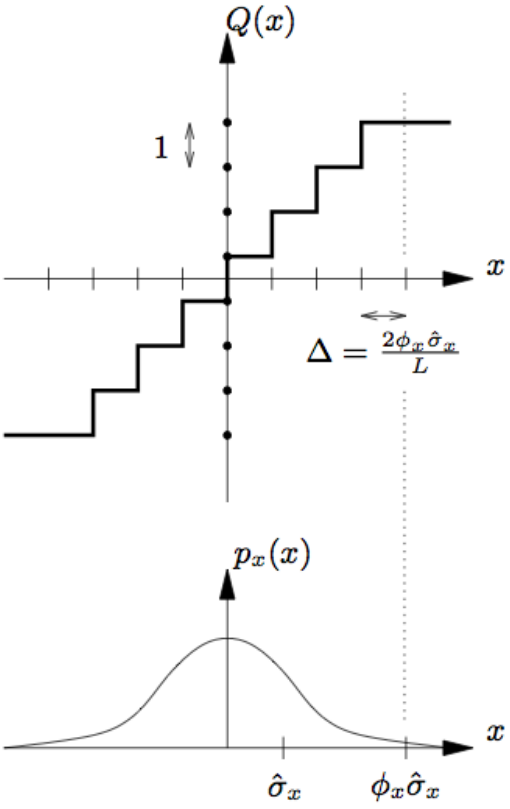
Adaptive Quantization

Motivated by the practical problem of non-stationary sources, adaptation of the uniform quantizer's stepsize is discussed. In particular, adaptive quantization based on forward estimation (AQF) and backward estimation (AQB) are discussed, in both block-based and recursive forms.

- Previously have considered the case of stationary source processes, though in reality the source signal may be highly non-stationary. For example, the variance, pdf, and/or mean may vary significantly with time.
- Here we concentrate on the problem of adapting uniform quantizer stepsize Δ to a signal with unknown variance. This is accomplished by estimating the input variance $\widehat{\sigma}_x\,(n)$ and setting the quantizer stepsize appropriately:
  Equation:

$$\Delta\,(n) = \frac{2\varphi_x\,\widehat{\sigma}_x\,(n)}{L}.$$

  Here φ_x is a constant that depends on the distribution of the input signal x whose function is to prevent input values greater than $\sigma_x\,(n)$ from being clipped by the quantizer (see [link]); comparing to non-adaptive step size relation $\Delta = 2x_{\max}/L$, we see that $\varphi_x\widehat{\sigma}_x\,(n) \sim x_{\max}$.

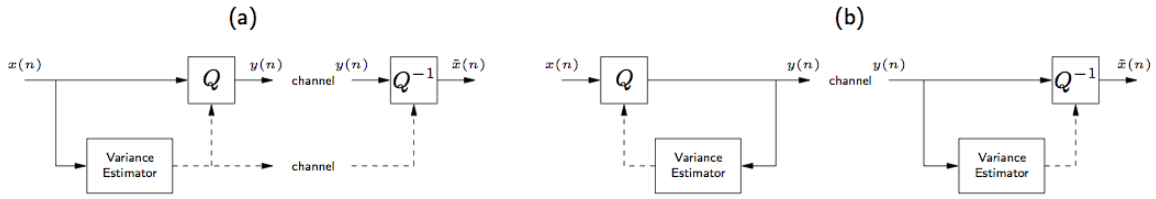


Adaptive quantization stepsize
$\delta\,(n) = 2\varphi_x\widehat{\sigma}_x/L$

- As long as the reconstruction levels $\{y_k\}$ are the same at encoder and decoder, the actual values chosen for quantizer design are arbitrary. Assuming integer values as in [link], the quantization rule becomes
  Equation:

$$y(n) = \begin{cases} \left\lceil \frac{x(n)}{\Delta(n)} \right\rceil - \frac{1}{2} & \text{midrise,} \\ \left\lceil \frac{x(n)}{\Delta(n)} - \frac{1}{2} \right\rceil & \text{midtread.} \end{cases}$$

- *AQF and AQB:* [link] shows two structures for stepsize adaptation: (a) *adaptive quantization with forward estimation* (AQF) and (b) *adaptive quantization with backward estimation* (AQB). The advantage of AQF is that variance estimation may be accomplished more accurately, as it is operates directly on the source as opposed to a quantized (noisy) version of the source. The advantage of AQB is that the variance estimates do not need to be transmitted as side information for decoding. In fact, practical AQF encoders transmit variance estimates only occasionally, e.g., once per block.



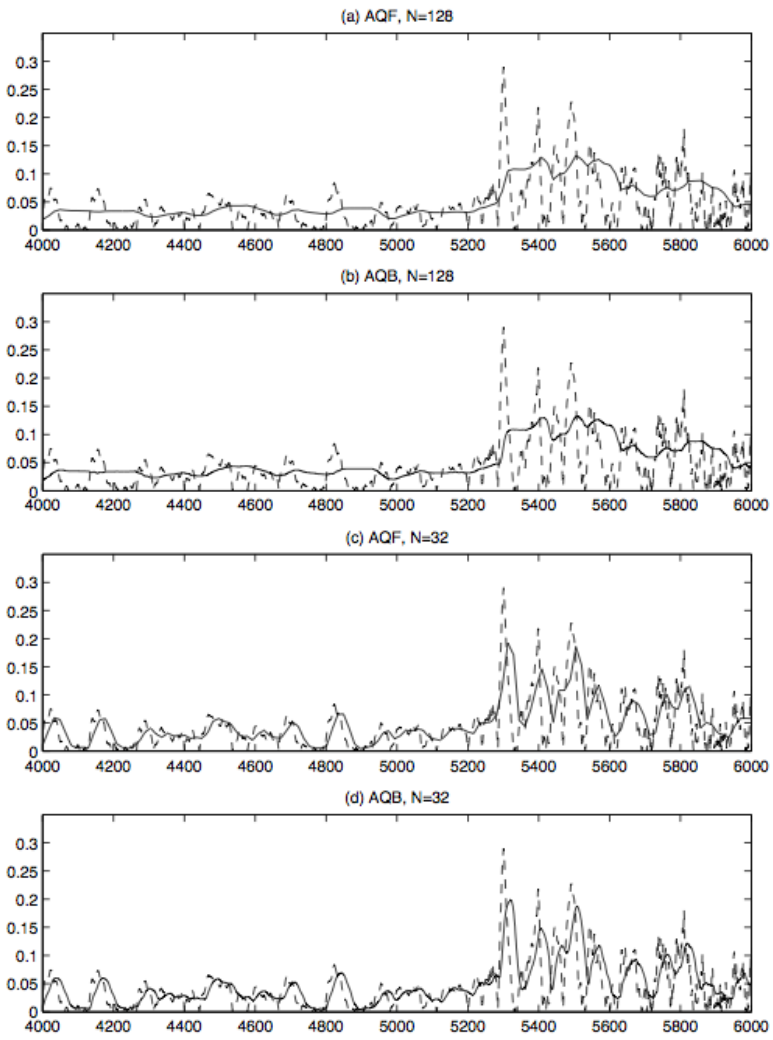
(a) AQF and (b) AQB

- *Block Variance Estimation:* When operating on finite blocks of data, the structures in [link] perform variance estimation as follows:
  Equation:

$$\text{Block AQF:} \quad \widehat{\sigma}_x^2(n) = \frac{1}{N} \sum_{i=1}^{N} x^2(n-i)$$

$$\text{Block AQB:} \quad \widehat{\sigma}_x^2(n) = \frac{1}{N} \sum_{i=1}^{N} \left(y(n-i) \cdot \Delta(n-i) \right)^2$$

  N is termed the *learning period* and its choice may significantly impact quantizer SNR performance: choosing N too large prevents the quantizer from adapting to the local statistics of the input, while choosing N too small results in overly noisy AQB variance estimates and excessive AQF side information. [link] demonstrates these two schemes for two choices of N.



Block AQF and AQB estimates of $\sigma_x(n)$ superimposed on $|x(n)|$ for $N = 128, 32$. SNR acheived: (a) 22.6 dB, (b) 28.8 dB, (c) 21.2 dB, and (d) 28.8 dB.

- *Recursive Variance Estimation:* The recursive method of estimating variance is as follows
  Equation:

$$\begin{aligned} \text{Recursive} \quad \text{AQF:} \quad \widehat{\sigma}_x^2(n) &= \alpha\widehat{\sigma}_x^2(n-1) + (1-\alpha)x^2(n-1) \\ \text{Recursive} \quad \text{AQB:} \quad \widehat{\sigma}_x^2(n) &= \alpha\widehat{\sigma}_x^2(n-1) + (1-\alpha)(y(n-1)\cdot\Delta(n-1))^2. \end{aligned}$$

  where α is a *forgetting factor* in the range $0 < \alpha < 1$ and typically near to 1. This leads to an exponential data window, as can be seen below. Plugging the expression for $\widehat{\sigma}_x^2(n-1)$ into that for $\widehat{\sigma}_x^2(n)$,
  Equation:

$$\begin{aligned}\widehat{\sigma}_x^2(n) &= \alpha\big(\alpha\widehat{\sigma}_x^2(n-2) + (1-\alpha)x^2(n-2)\big) + (1-\alpha)x^2(n-1) \\ &= \alpha^2\widehat{\sigma}_x^2(n-2) + (1-\alpha)\big(x^2(n-1) + \alpha x^2(n-2)\big).\end{aligned}$$

Then plugging $\widehat{\sigma}_x^2(n-2)$ into the above,
Equation:

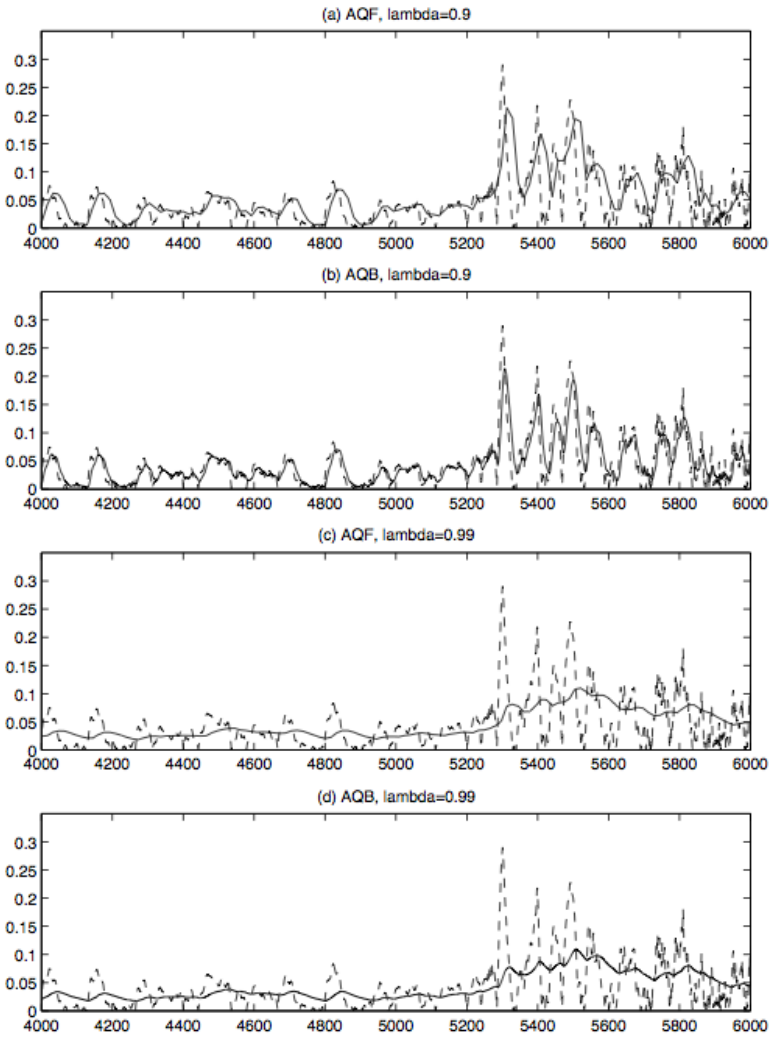
$$\begin{aligned}\widehat{\sigma}_x^2(n) &= \alpha^2\big(\alpha\widehat{\sigma}_x^2(n-3) + (1-\alpha)x^2(n-3)\big) + (1-\alpha)\big(x^2(n-1) + \alpha x^2(n-2)\big) \\ &= \alpha^3\widehat{\sigma}_x^2(n-3) + (1-\alpha)\big(x^2(n-1) + \alpha x^2(n-2) + \alpha^2 x^2(n-3)\big).\end{aligned}$$

Continuing this process *N* times, we arrive at
Equation:

$$\widehat{\sigma}_x^2(n) = (1-\alpha)\sum_{i=1}^{N}\alpha^{i-1}x^2(n-i) + \alpha^N\widehat{\sigma}_x^2(n-N).$$

Taking the limit as $N \to \infty$, $\alpha < 1$ ensures that
Equation:

$$\boxed{\widehat{\sigma}_x^2(n) = (1-\alpha)\sum_{i=1}^{\infty}\alpha^{i-1}x^2(n-i).}$$



Exponential AQF and AQB estimates of $\sigma_x\,(n)$ superimposed on $|x(n)|$ for $\lambda = 0.9, 0.99$. (a) 20.5 dB, (b) 28.0 dB, (c) 22.2 dB, (d) 24.1 dB.

Pulse Code Modulation

This module prefaces the subsequent modules on differential pulse code modulation (DPCM). In this module, the standard technique known as pulse code modulation (PCM) is described.

- PCM is the “standard” sampling method taught in introductory DSP courses. The input signal is

  1. filtered to prevent aliasing,
  2. discretized in time by a sampler, and
  3. quantized in amplitude by a quantizer

  before transmission (or storage). Finally, the received samples are interpolated in time and amplitude to reconstruct an approximation of the input signal. Note that transmission may employ the use of additional encoding and decoding, as with entropy codes. PCM is the most widespread and well-understood digital coding system for reasons of simplicity, though not efficiency (as we shall see).

Standard PCM system

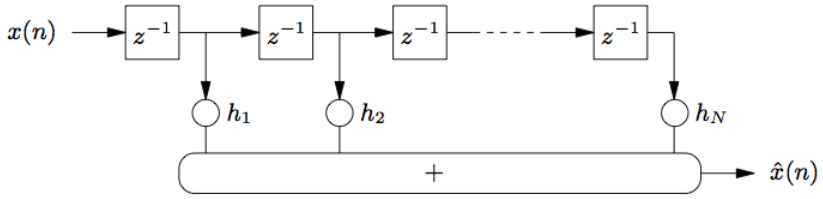
Differential Pulse Code Modulation

Differential pulse code modulation (DPCM) is described. First, quantized predictive encoding is motivated but then shown to suffer from amplification of quantization error at the decoder. This problem is avoided by DPCM, which places the quantizer in the prediction loop.

- Many information signals, including audio, exhibit significant redundancy between successive samples. In these situations, it is advantageous to transmit only the difference between predicted and true versions of the signal: with a "good" predictions, the quantizer input will have variance less than the original signal, allowing a quantizer with smaller decision regions and hence higher SNR. (See [link] for an example of such a structure.)
- *Linear Prediction:* There are various methods of prediction, but we focus on *forward linear prediction of order N*, illustrated by [link] and described by the following equation, where $\widehat{x}\,(n)$ is a linear estimate of $x(n)$ based on N previous versions of $x(n)$:
  Equation:

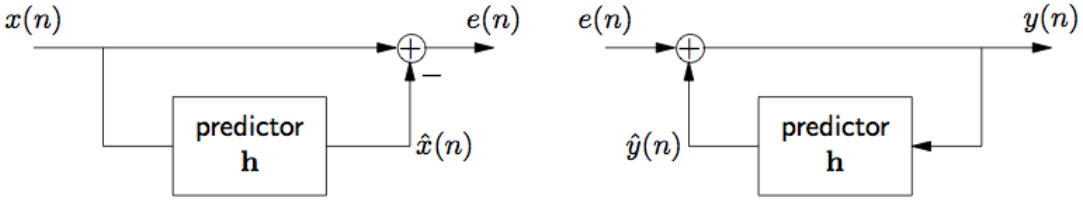
$$\widehat{x}\,(n) \;=\; \sum_{i=1}^{N} h_i x\,(n-i).$$

  It will be convenient to collect the *prediction coefficients* into the vector $\mathbf{h} = (h_1, h_2, \cdots, h_N)^t$.



Linear Prediction

- *Lossless Predictive Encoding:* Consider first the system in [link].



Lossless Predictive Data Transmission System

The system equations are
Equation:

$$e\left(n\right)=x\left(n\right)-\widehat{x}\left(n\right)$$
$$y\left(n\right)=e\left(n\right)+\hat{y}\left(n\right)$$

In the *z*-domain (i.e., $X\left(z\right)=\sum_{n}x\left(n\right)z^{-n}$ and $H\left(z\right)=\sum_{i}h_{i}z^{-i}$),
Equation:

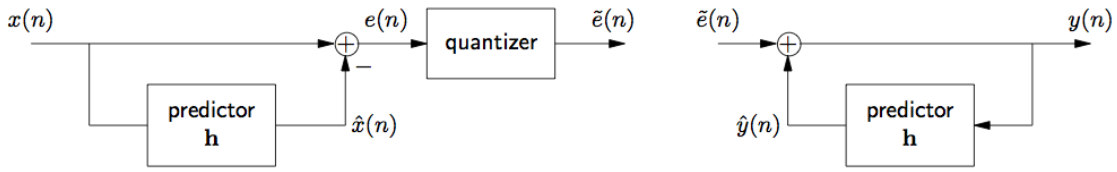
$$\begin{aligned}E(z) &= X\left(z\right)-\widehat{X}\left(z\right)=X\left(z\right)(1-H(z))\\ Y(z) &= E\left(z\right)+\widehat{Y}\left(z\right)=E\left(z\right)+H\left(z\right)Y\left(z\right)\end{aligned}$$

We call this transmission system lossless because, from above,
Equation:

$$Y\left(z\right)=\frac{E(z)}{1-H(z)}=X\left(z\right)$$

Without quantization, however, the prediction error $e(n)$ takes on a continuous range of values, and so this scheme is not applicable to digital transmission.

- *Quantized Predictive Encoding:* Quantizing the prediction error in [link], we get the system of [link].



Quantized Predictive Coding System

Here the equations are
Equation:

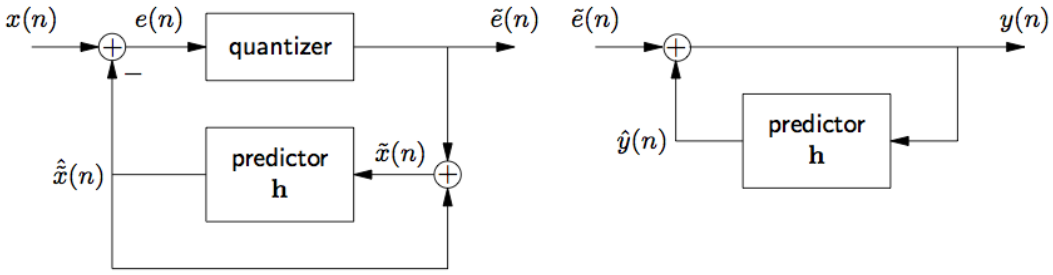
$$\begin{aligned} q(n) &= \tilde{e}\,(n) - e\,(n) \\ e(n) &= x\,(n) - \widehat{x}\,(n) \\ y(n) &= \tilde{e}\,(n) + \hat{y}\,(n) \end{aligned}$$

In the z-domain we find that
Equation:

$$\begin{aligned} \widetilde{E}\,(z) &= X\,(z)(1 - H(z)) + Q\,(z) \\ Y(z) &= \frac{\widetilde{E}\,(z)}{1 - H(z)} = X\,(z) + \frac{Q(z)}{1 - H(z)}. \end{aligned}$$

Thus the reconstructed output is corrupted by a filtered version of the quantization error where the filter $(1 - H(z))^{-1}$ is expected to amplify the quantization error; recall that $Y\,(z) = E\,(z)(1 - H(z))^{-1}$ where the goal of prediction was to make $\sigma_e^2 < \sigma_y^2$. This problem results from the fact that the quantization noise appears at the decoder's predictor input but not at the encoder's predictor input. But we can avoid this...

- *DPCM:* Including quantization in the encoder's prediction loop, we obtain the system in [link], known as *differential pulse code modulation* .



A Typical Differential PCM System

System equations are
Equation:

$$\begin{aligned} q(n) &= \tilde{e}\,(n) - e\,(n) \\ e(n) &= x\,(n) - \widehat{\widetilde{x}}\,(n) \\ \widetilde{x}\,(n) &= \tilde{e}\,(n) + \widehat{\widetilde{x}}\,(n) \\ y(n) &= \tilde{e}\,(n) + \hat{y}\,(n). \end{aligned}$$

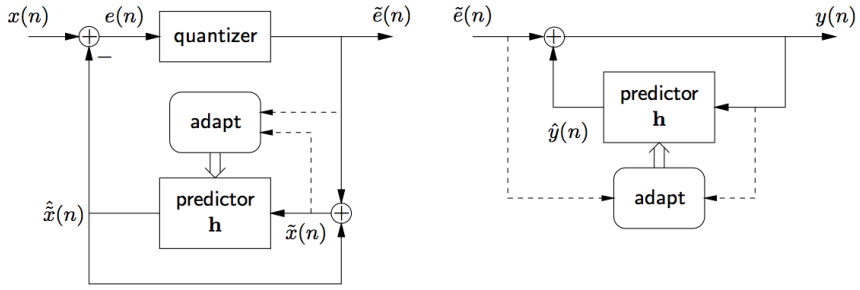
In the z-domain we find that
Equation:

$$\begin{aligned}\widetilde{E}(z) &= X(z) - H(z)\widetilde{X}(z) + Q(z) \\ \widetilde{X}(z) &= (Q(z) + E(z)) + (X(z) - E(z)) = X(z) + Q(z) \\ Y(z) &= \frac{\widetilde{E}(z)}{1 - H(z)}\end{aligned}$$

so that
Equation:

$$Y(z) = \frac{X(z) - H(z)(X(z) + Q(z)) + Q(z)}{1 - H(z)} = X(z) + Q(z) = \widetilde{X}(z).$$

Thus, the reconstructed output is corrupted only by the quantization error. Another significant advantage to placing the quantizer inside the prediction loop is realized if the predictor made self-adaptive (in the same spirit as the adaptive quantizers we studied). As illustrated in [link], adaptation of the prediction coefficients can take place simulateously at the encoder and decoder with no transmission of side-information (e.g. $\mathbf{h}(n)$)! This is a consequence of the fact that both algorithms have access to identical signals.



Adaptive DPCM System

Performance of DPCM
Here we characterize the performance of DPCM via the simpler surrogate known as "quantized predictive encoding", which is known to have very similar performance in practice. To do this, we derive the optimum prediction coefficients, the resulting prediction error variance and gain over PCM.

- As we noted earlier, the DPCM performance gain is a consequence of variance reduction obtained through prediction. Here we derive the optimal predictor coefficients, prediction error variance, and bit rate for the system in figure 4 from Differential Pulse Code Modulation. This system is easier to analyze than DPCM systems with quantizer in loop (e.g., figure 5 from Differential Pulse Code Modulation) and it is said that the difference in prediction-error behavior is negligible when $R > 2$(see page 267 of Jayant & Noll).
- *Optimal Prediction Coefficients:* First we find coefficients **h** minimizing prediction error variance:
  Equation:

$$\min_{\mathbf{h}} \mathrm{E}\left\{e^2(n)\right\}.$$

  Throughout, we assume that $x(n)$ is a zero-mean stationary random process with autocorrelation
  Equation:

$$r_x(k) := \mathrm{E}\left\{x(n)x(n-k)\right\} = r_x(-k).$$

  A necessary condition for optimality is the following:
  Equation:

$$\begin{aligned}
\forall j \in \{1, \cdots, N\}, \quad 0 &= \frac{1}{2}\frac{\partial}{\partial h_j}\mathrm{E}\left\{e^2(n)\right\} \\
&= \mathrm{E}\left\{e(n)\frac{\partial e(n)}{\partial h_j}\right\} \\
&= \mathrm{E}\left\{e(n)x(n-j)\right\} \qquad \leftarrow \mathrm{The"Orthogonality Principle"} \\
&= \mathrm{E}\left\{\left(x(n) - \sum_{i=1}^{N} h_i x(n-i)\right)x(n-j)\right\} \\
&= \mathrm{E}\left\{x(n)x(n-j)\right\} - \sum_{i=1}^{N} h_i \,\mathrm{E}\left\{x(n-i)x(n-j)\right\} \\
&= r_x(j) - \sum_{i=1}^{N} h_i r_x(j-i)
\end{aligned}$$

  where we have used equation 1 from Differential Pulse Code Modulation. We can rewrite this as a system of linear equations:
  Equation:

$$\underbrace{\begin{pmatrix} r_x(1) \\ r_x(2) \\ \vdots \\ r_x(N) \end{pmatrix}}_{\mathbf{r}_x} = \underbrace{\begin{pmatrix} r_x(0) & r_x(1) & \cdots & r_x(N-1) \\ r_x(1) & r_x(0) & \cdots & r_x(N-2) \\ \vdots & & & \\ r_x(N-1) & r_x(N-2) & \cdots & r_x(0) \end{pmatrix}}_{\mathbf{R}_N} \underbrace{\begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_N \end{pmatrix}}_{\mathbf{h}}$$

which yields an expression for the optimal prediction coefficients:

Equation:

$$\mathbf{h}_\star = \mathbf{R}_N^{-1}\mathbf{r}_x.$$

- *Error for Length-N Predictor:* The definition $\mathbf{x}(n) := (x(n), x(n-1), \cdots, x(n-N))^t$ and [link] can be used to show that the minimum prediction error variance is
  Equation:

$$\begin{aligned} \sigma_e^2\big|_{\min,N} &= \mathrm{E}\left\{e^2(n)\right\} \\ &= \mathrm{E}\left\{\left\|\mathbf{x}^t(n)\begin{pmatrix} 1 \\ -\mathbf{h}_\star \end{pmatrix}\right\|^2\right\} \\ &= \begin{pmatrix} 1 & -\mathbf{h}_\star^t \end{pmatrix} \mathrm{E}\left\{\mathbf{x}(n)\mathbf{x}^t(n)\right\}\begin{pmatrix} 1 \\ -\mathbf{h}_\star \end{pmatrix} \\ &= \begin{pmatrix} 1 & -\mathbf{h}_\star^t \end{pmatrix}\begin{pmatrix} r_x(0) & \mathbf{r}_x^t \\ \mathbf{r}_x & \mathbf{R}_N \end{pmatrix}\begin{pmatrix} 1 \\ -\mathbf{h}_\star \end{pmatrix} \\ &= r_x(0) - 2\mathbf{h}_\star^t\mathbf{r}_x + \mathbf{h}_\star^t\mathbf{R}_N\mathbf{h}_\star \\ &= r_x(0) - \mathbf{r}_x^t\mathbf{R}_N^{-1}\mathbf{r}_x. \end{aligned}$$

- *Error for Infinite-Length Predictor:* We now characterize $\sigma_e^2\big|_{\min,N}$ as $N \to \infty$. Note that
  Equation:

$$\underbrace{\begin{pmatrix} r_x(0) & \mathbf{r}_x^t \\ \mathbf{r}_x & \mathbf{R}_N \end{pmatrix}}_{\mathbf{R}_{N+1}}\begin{pmatrix} 1 \\ -\mathbf{h}_\star \end{pmatrix} = \begin{pmatrix} \sigma_e^2\big|_{\min,N} \\ \mathbf{0} \end{pmatrix}$$

  Using Cramer's rule,
  Equation:

$$1 = \frac{\begin{vmatrix} \sigma_e^2|_{\min,N} & \mathbf{r}_x^t \\ \mathbf{0} & \mathbf{R}_N \end{vmatrix}}{|\mathbf{R}_{N+1}|} = \sigma_e^2\Big|_{\min,N} \frac{|\mathbf{R}_N|}{|\mathbf{R}_{N+1}|} \quad \Rightarrow \quad \sigma_e^2\Big|_{\min,N} = \frac{|\mathbf{R}_{N+1}|}{|\mathbf{R}_N|}.$$

Note:
Cramer's Rule
Given matrix equation $\mathbf{A}\mathbf{y} = \mathbf{b}$, where $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_N) \in \mathbb{R}^{N\times N}$,
Equation:

$$y_k = \frac{|\mathbf{a}_1, \cdots, \mathbf{a}_{k-1}, \mathbf{b}, \mathbf{a}_{k+1}, \cdots, \mathbf{a}_N|}{|\mathbf{A}|}$$

where $|\cdot|$ denotes determinant.

A result from the theory of Toeplitz determinants (see Jayant & Noll) gives the final answer:
Equation:

$$\sigma_e^2|_{\min} = \lim_{N\to\infty} \frac{|\mathbf{R}_{N+1}|}{|\mathbf{R}_N|} = \boxed{\exp\left(\frac{1}{2\pi}\int_{-\pi}^{\pi} \ln S_x\left(e^{j\omega}\right) d\omega\right)}$$

where $S_x\left(e^{j\omega}\right)$ is the *power spectral density* of the WSS random process $x(n)$:
Equation:

$$S_x\left(e^{j\omega}\right) := \sum_{n=-\infty}^{\infty} r_x(n) e^{-j\omega n}.$$

(Note that, because $r_x(n)$ is conjugate symmetric for stationary $x(n)$, $S_x\left(e^{j\omega}\right)$ will always be non-negative and real.)

- *ARMA Source Model:* If the random process $x(n)$ can be modelled as a *general linear process* , i.e., white noise $v(n)$ driving a causal LTI system $B(z)$:
  Equation:

$$x(n) = v(n) + \sum_{k=1}^{\infty} b_k v(n-k) \quad \text{with} \quad \sum_k |b_k|^2 < \infty,$$

then it can be shown that
Equation:

$$\sigma_v^2 \;= \exp\left(\frac{1}{2\pi}\int_{-\pi}^{\pi} \ln\, S_x\left(e^{j\omega}\right) d\omega\right).$$

Thus the MSE-optimal prediction error variance equals that of the driving noise $v(n)$ when $N = \infty$.

- *Prediction Error Whiteness:* We can also demonstrate that the MSE-optimal prediction error is white when $N = \infty$. This is a simple fact of the orthogonality principle seen earlier:
  Equation:

$$\boxed{0 \;= \mathrm{E}\left\{e(n)x(n-k)\right\}, \quad k = 1, 2, \cdots}.$$

  The prediction error has autocorrelation
  Equation:

$$\begin{aligned}
\mathrm{E}\left\{e(n)e(n-k)\right\} &= \mathrm{E}\left\{e\left(n\right)\left(x\left(n-k\right) + \sum_{i=1}^{\infty} h_i x\left(n-k-i\right)\right)\right\} \\
&= \underbrace{\mathrm{E}\left\{e(n)x(n-k)\right\}}_{\to 0 \,\text{for}\, k>0} + \sum_{i=1}^{\infty} h_i \underbrace{\mathrm{E}\left\{e(n)x(n-k-i)\right\}}_{\to 0} \\
&= \sigma_e^2|_{\min}\delta\left(k\right).
\end{aligned}$$

- *AR Source Model:* When the input can be modelled as an autoregressive (AR) process of order *N*:
  Equation:

$$X\left(z\right) \;=\; \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}} V\left(z\right),$$

  then MSE-optimal results (i.e., $\sigma_e^2 = \sigma_e^2\big|_{\min}$ and whitening) may be obtained with a forward predictor of order *N*. Specifically, the prediction coefficients h_i can be chosen as $h_i = a_i$ and so the prediction error $E(z)$ becomes
  Equation:

$$E\left(z\right) \;=\; \left(1 - H(z)\right)X\left(z\right) \;=\; \frac{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}} V\left(z\right) \;=\; V\left(z\right),$$

- *Efficiency Gain over PCM:* Prediction reduces the variance at the quantizer input without changing the variance of the reconstructed signal.
  - By keeping the number of quantization levels fixed, could reduce quantization step width and obtain lower quantization error than PCM at the same bit rate.
  - By keeping the decision levels fixed, could reduce the number of quantization levels and obtain a lower bit rate than PCM at the same quantization error level.

Assuming that $x(n)$ and $e(n)$ are distributed similarly, use of the same style of quantizer on DPCM vs. PCM systems yields
Equation:

$$\mathrm{SNR}_{\mathrm{DPCM}} \;=\; \mathrm{SNR}_{\mathrm{PCM}} + 10\,\log_{10}\,\frac{\sigma_x^2}{\sigma_e^2}.$$

Analysis of DPCM using Rate-Distortion Theory
Using rate-distortion theory, the optimal SNR attainable for rate-R source coding is related to R and the spectral flatness measure of the source. The SNR of rate-R DPCM is then analyzed and compared to the optimal, and shown to suffer by only 1.53 dB.

- The *rate-distortion function*$R(D)$ specifies the minimum average rate *R* required to transmit the source process at a mean distortion of *D*, while the *distortion-rate function*$D(R)$ specifies the minimum mean distortion *D* resulting from transmission of the source at average rate *R*. These bounds are theoretical in the sense that coding techniques which attain these minimum rates or distortions are in general unknown and thought to be infinitely complex as well as require infinite memory. Still, these bounds form a reference against which any specific coding system can be compared. For a continuous-amplitude white (i.e., "memoryless") Gaussian source $x(n)$ (see Berger and Jayant & Noll),
  Equation:

$$\begin{aligned} R(D) &= \begin{cases} \frac{1}{2}\log_2 \frac{\sigma_x^2}{D} & 0 \le D \le \sigma_x^2 \\ 0 & D \ge \sigma_x^2 \end{cases} \\ D(R) &= 2^{-2R}\sigma_x^2. \end{aligned}$$

  The sources we are interested in, however, are non-white. It turns out that when distortion *D* is "small," non-white Gaussian $x(n)$ have the following distortion-rate function: (see page 644 of Jayant & Noll)
  Equation:

$$\begin{aligned} D(R) &= 2^{-2R}\exp\left(\frac{1}{2\pi}\int_{-\pi}^{\pi}\ln S_x\left(e^{j\omega}\right)d\omega\right) \\ &= 2^{-2R}\sigma_x^2\underbrace{\left(\frac{\exp\left(\frac{1}{2\pi}\int_{-\pi}^{\pi}\ln S_x\left(e^{j\omega}\right)d\omega\right)}{\frac{1}{2\pi}\int_{-\pi}^{\pi}S_x\left(e^{j\omega}\right)d\omega}\right)}_{\text{spectral flatness measure}}. \end{aligned}$$

Note the ratio of geometric to arithmetic PSD means, called the *spectral flatness measure*. Thus optimal coding of $x(n)$ yields
Equation:

$$\begin{aligned} \text{SNR}(R) &= 10\log_{10}\left(\frac{\sigma_x^2}{D(R)}\right) \\ &\approx \boxed{6.02R - 10\log_{10}\left(\text{SFM}_x\right)}. \end{aligned}$$

To summarize, [link] (lower equation) gives the best possible SNR for *any* arbitrarily-complex coding system that transmits/stores information at an average rate of *R* bits/sample.

- Let's compare the SNR-versus-rate performance acheivable by DPCM to the optimal given by [link] (lower equation). The structure we consider is shown in [link], where quantized DPCM outputs $\tilde{e}\,(n)$ are coded into binary bits using an entropy coder. Assuming that $\tilde{e}\,(n)$ is white (which is a good assumption for well-designed predictors), optimal entropy coding/decoding is able to transmit and recover $\tilde{e}\,(n)$ at $R = H_{\tilde{e}}$ bits/sample without any distortion. $H_{\tilde{e}}$ is the entropy of $\tilde{e}\,(n)$, for which we derived the following expression assuming large-*L* uniform quantizer:
Equation:

$$H_{\tilde{e}} = h_e - \frac{1}{2}\log_2\left(12\text{var}(e\,(n) - \tilde{e}\,(n))\right).$$

Since var $(e\,(n) - \tilde{e}\,(n)) = \sigma_r^2$ in DPCM, $H_{\tilde{e}}$ can be rewritten:
Equation:

$$H_{\tilde{e}} = h_e - \frac{1}{2}\log_2\left(12\sigma_r^2\right).$$

If $e(n)$ is Gaussian, it can be shown that the differential entropy h_e takes on the value
Equation:

$$h_e = \frac{1}{2}\log_2\left(2\pi e\sigma_e^2\right),$$

so that
Equation:

$$H_{\tilde{e}} \quad = \quad \frac{1}{2}\log_2\left(\frac{\pi e \sigma_e^2}{6\sigma_r^2}\right).$$

Using $R = H_{\tilde{e}}$ and rearranging the previous expression, we find
Equation:

$$\sigma_r^2 \quad = \quad \frac{\pi e}{6} 2^{-2R}\sigma_e^2.$$

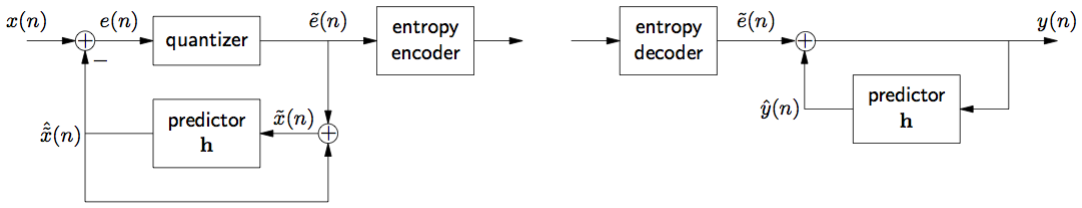
With the optimal infinite length predictor, ${\sigma_e}^2$ equals $\sigma_e^2\big|_{\min}$ given by equation 10 from Performance of DPCM. Plugging equation 10 from Performance of DPCM into the previous expression and writing the result in terms of the spectral flatness measure,
Equation:

$$\sigma_r^2 \quad = \quad \frac{\pi e}{6} 2^{-2R}\sigma_x^2\,\mathrm{SFM}_x.$$

Translating into SNR, we obtain
Equation:

$$\begin{aligned}\mathrm{SNR} \quad &= \quad 10\log_{10}\left(\frac{\sigma_x^2}{\sigma_r^2}\right) \\ &\approx \quad \boxed{6.02R - 1.53 - 10\log_{10}\mathrm{SFM}_x} \quad [\mathrm{dB}].\end{aligned}$$

To summarize, a DPCM system using a MSE-optimal infinite-length predictor and optimal entropy coding of $\tilde{e}\,(n)$ could operate at an average of *R* bits/sample with the SNR in [link] (lower equation).

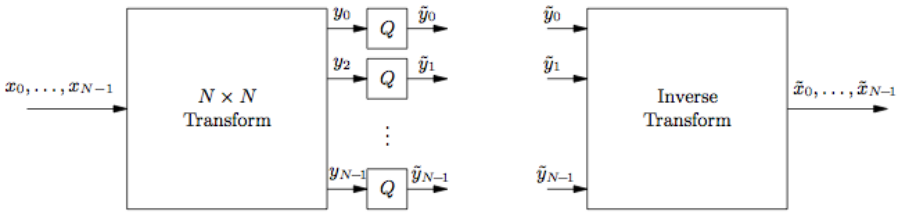


Entropy-Encoded DPCM System.

- Comparing [link] (lower equation) and [link] (lower equation), we see that DPCM incurs a 1.5 dB penalty in SNR when compared to the optimal. From our previous discussion on optimal quantization, we recognize that this 1.5 dB penalty comes from the fact that the quantizer in the DPCM system is memoryless. (Note that the DPCM quantizer *must* be memoryless since the predictor input must not be delayed.)
- Though we have identified a 1.5 dB DPCM penalty with respect to optimal, a key point to keep in mind is that the design of near-optimal coders for *non-white* signals is extremely difficult. When the signal statistics are rapidly changing, such a design task becomes nearly impossible. Though still non-trivial to design, near-optimal entropy coders for *white* signals exist and are widely used in practice. Thus, DPCM can be thought of as a way of pre-processing a colored signal that makes near-optimal coding possible. From this viewpoint, 1.5 dB might not be considered a high price to pay.

Transform Coding: Background and Motivation
Transform coding is described and an analysis is performed for the simple 2-dimensional case, including a comparison to PCM.

- In transform coding (TC), blocks of *N* input samples are transformed to *N* transform coefficients which are then quantized and transmitted. At the decoder, an inverse transform is applied to the quantized coefficients, yielding a reconstruction of the original waveform. By designing individual quantizers in accordance with the statistics of their inputs, it is possible to allocate bits in a more optimal manner, e.g., encoding the "more important" coefficients at a higher bit rate.



$N \times N$ Transform Coder/Decoder with Scalar Quantization

- *Orthogonal Transforms:* From our perspective, an $N \times N$ "transform" will be any real-valued linear operation taking *N* input samples to *N* output samples, or transform coefficients. This operation can always be written in matrix form
Equation:

$$\mathbf{y}(m) = \mathbf{T}\mathbf{x}(m), \quad \mathbf{T} \in \mathbb{R}^{N \times N}$$

where $\mathbf{x}(m)$ and $\mathbf{y}(m)$ are vectors representing $N \times 1$ blocks of input/output elements:
Equation:

$$\begin{aligned} \mathbf{x}(m) &= (x(mN),\ x(mN-1),\ \cdots,\ x(mN-N+1))^t \\ \mathbf{y}(m) &= (y(mN),\ y(mN-1),\ \cdots,\ y(mN-N+1))^t. \end{aligned}$$

Intuition comes from considering the transform's *basis vectors* $\{\mathbf{t}_k\}$ defined by the rows of the matrix
Equation:

$$\mathbf{T} = \begin{matrix} \text{--- } \mathbf{t}_0^t \text{ ---} \\ \text{--- } \mathbf{t}_1^t \text{ ---} \\ \vdots \\ \text{-- } \mathbf{t}_{N-1}^t \text{ --} \end{matrix}$$

since the coefficient $y_k = \mathbf{t}_k^t \mathbf{x}$ can be thought of as the result of a "comparison" between the k^{th} basis vector and the input $\mathbf{x}$. These comparisons are defined by the inner product $<\mathbf{t}_k, \mathbf{x}> = \mathbf{t}_k^t \mathbf{x}$ which has a geometrical interpretation involving the angle θ_k between vectors $\mathbf{t}_k$ and $\mathbf{x}$.
Equation:

$$<\mathbf{t}_k, \mathbf{x}> = \cos(\theta_k)\ \|\ \mathbf{t}_k\ \|_2\ \|\ \mathbf{x}\ \|_2.$$

When the vectors $\{\mathbf{t}_k\}$ are mutually orthogonal, i.e., $\mathbf{t}_k^t \mathbf{t}_\ell = 0$ for $k \neq \ell$, the transform coefficients represent separate, unrelated features of the input. This property is convenient if the transform

coefficients are independently quantized, as is typical in TC schemes.

Example:
2×2 Transform Coder
Say that stationary zero-mean Gaussian source $x(m)$ has autocorrelation $r_x(0) = 1$, $r_x(1) = \rho$, and $r_x(k) = 0$ for $k > 1$. For a bit rate of *R* bits per sample, uniformly-quantized PCM implies a mean-squared reconstruction error of
Equation:

$$\underset{\text{PCM}}{\sigma_r^2} = \underset{\substack{\Delta = 2x_{\max}/L \\ L = 2^R}}{\frac{\Delta^2}{12}} = \underset{\gamma_x}{\frac{1}{3}\frac{x_{\max}^2}{\sigma_x^2}}\sigma_x^2 2^{-2R} = \gamma_x \sigma_x^2 2^{-2R}.$$

For transform coding, say we choose linear transform
Equation:

$$\mathbf{T} = \begin{pmatrix} \mathbf{t}_0^t \\ \mathbf{t}_1^t \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Setting $\mathbf{x}(m) = (x(2m) \quad x(2m-1))^t$ and $\mathbf{y}(m) = \mathbf{T}\mathbf{x}(m)$, we find that the transformed coefficients have variance
Equation:

$$\sigma_{y_0}^2 = \mathrm{E}\left\{|\mathbf{t}_0^t \mathbf{x}(m)|^2\right\} = \frac{1}{2}\mathrm{E}\left\{|x(2m) + x(2m-1)|^2\right\} = \frac{1}{2}(2r_x(0) + 2r_x(1)) = 1 + \rho$$

Equation:

$$\sigma_{y_1}^2 = \mathrm{E}\left\{|\mathbf{t}_1^t \mathbf{x}(m)|^2\right\} = \frac{1}{2}\mathrm{E}\left\{|x(2m) - x(2m-1)|^2\right\} = \frac{1}{2}(2r_x(0) - 2r_x(1)) = 1 - \rho$$

and using uniformly-quantized PCM on each coefficient we get mean-squared reconstruction errors
Equation:

$$\sigma_{q_0}^2 = (1+\rho)\gamma_x 2^{-2R_0}$$

Equation:

$$\sigma_{q_1}^2 = (1-\rho)\gamma_x 2^{-2R_1}.$$

We use the same quantizer performance factor γ_x as before since linear operations preserve Gaussianity. For orthogonal matrices $\mathbf{T}$, i.e., $\mathbf{T}^{-1} = \mathbf{T}^t$, we can show that the mean-squared reconstruction error σ_r^2 equals the mean-squared quantization error:
Equation:

$$
\begin{aligned}
\sigma_r^2 &:= \frac{1}{N}\sum_{k=0}^{N-1} \mathrm{E}\left\{\left(\widetilde{x}\left(Nm-k\right)-x\left(Nm-k\right)\right)^2\right\} \qquad (\text{here } N=2)\\
&= \frac{1}{N}\,\mathrm{E}\left\{\parallel \widetilde{\mathbf{x}}(m)-\mathbf{x}\left(m\right)\parallel^2\right\}\\
&= \frac{1}{N}\,\mathrm{E}\left\{\parallel \mathbf{T}^{-1}\widetilde{\mathbf{y}}(m)-\mathbf{x}\left(m\right)\parallel^2\right\}\\
&= \frac{1}{N}\,\mathrm{E}\left\{\parallel \mathbf{T}^{-1}(\mathbf{y}(m)+\mathbf{q}(m))-\mathbf{x}\left(m\right)\parallel^2\right\}\\
&= \frac{1}{N}\,\mathrm{E}\left\{\parallel \mathbf{T}^{-1}\mathbf{T}\mathbf{x}\left(m\right)+\mathbf{T}^{-1}\mathbf{q}\left(m\right)-\mathbf{x}\left(m\right)\parallel^2\right\}\\
&= \frac{1}{N}\,\mathrm{E}\left\{\parallel \mathbf{T}^{-1}\mathbf{q}\left(m\right)\parallel^2\right\}\\
&= \frac{1}{N}\,\mathrm{E}\quad \underset{\mathbf{I}}{\mathbf{q}^t\left(m\right)\left(\mathbf{T}^{-1}\right)^t\mathbf{T}^{-1}}\mathbf{q}\left(m\right)\\
&= \frac{1}{N}\,\mathrm{E}\left\{\parallel \mathbf{q}\left(m\right)\parallel^2\right\}\\
&= \frac{1}{N}\sum_{k=0}^{N-1}\sigma_{q_k}^2.
\end{aligned}
$$

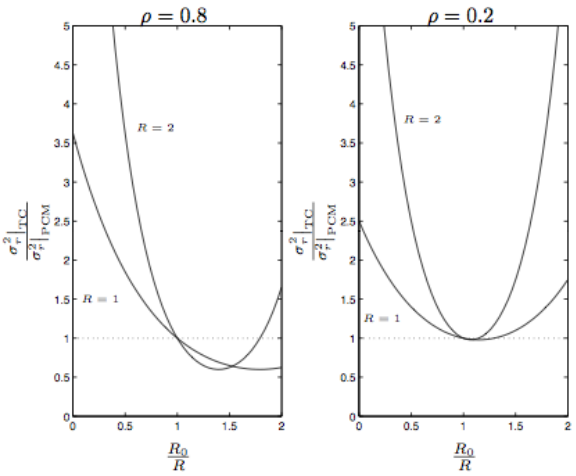
Since our 2×2 matrix is indeed orthogonal, we have mean-squared reconstruction error
Equation:

$$\sigma_r^2|_{\mathrm{TC}} = \frac{1}{2}\left((1+\rho)\gamma_x 2^{-2R_0}+(1-\rho)\gamma_x 2^{-2R_1}\right)$$

at bit rate of R_0+R_1 bits per two samples. Comparing TC to PCM at equal bit rates (i.e. $R_0+R_1=2R$),
Equation:

$$\frac{\sigma_r^2{}_{\mathrm{TC}}}{\sigma_r^2|_{\mathrm{PCM}}} = \frac{1}{2}\frac{(1+\rho)\gamma_x 2^{-2R_0}+(1-\rho)\gamma_x 2^{-2(2R-R_0)}}{\gamma_x 2^{-2R}} = (1+\rho)2^{2(R-R_0)-1}+(1-\rho)2^{2(R_0-R)-1}.$$

[link] shows that (i) allocating a higher bit rate to the quantizer with stronger input signal can reduce the average reconstruction error relative to PCM, and (ii) the gain over PCM is higher when the input signal exhibits stronger correlation ρ. Also note that when $R_0=R_1=R$, there is no gain over PCM—a verification of the fact that $\sigma_r^2=\sigma_q^2$ when $\mathbf{T}$ is orthogonal.



Ratio of TC to PCM mean-squared reconstruction errors versus bit rate R_0 for two values of ρ.

Optimal Bit Allocation

For transform coding with a fixed total bit rate, the optimal (SNR-maximizing) allocation of bit rates is derived.

- *Motivating Question:* Assuming that $\mathbf{T}$ is an $N \times N$ orthogonal matrix, what is the MSE-optimal bitrate allocation strategy assuming independent uniform quantization of the transform outputs? In other words, what $\{R_\ell\}$ minimize average reconstruction error for fixed average rate $\frac{1}{N}\sum_\ell R_\ell$?
- Say that the k^{th} element of the transformed output vector $\mathbf{y}(n)$ has variance $\left\{\sigma_{y_k}^2\right\}$. With uniform quantization, example 1 from Background and Motivation showed that the k^{th} quantizer error power is
  Equation:

$$\sigma_{q_k}^2 \;=\; \gamma_{y_k}\sigma_{y_k}^2 2^{-2R_k},$$

  where R_k is the bit rate allocated to the k^{th} quantizer output and where γ_{y_k} depends on the distribution of y_k. From this point on we make the simplifying assumption that γ_{y_k} is independent of k. As shown in example 1 from Background and Motivation, orthogonal matrices imply that the mean squared reconstruction error equals the mean squared quantization error, so that
  Equation:

$$\sigma_r^2 \;=\; \frac{1}{N}\sum_{k=0}^{N-1}\sigma_{q_k}^2 \;=\; \frac{\gamma_y}{N}\sum_{k=0}^{N-1}\sigma_{y_k}^2 2^{-2R_k}.$$

  Thus we have the constrained optimization problem
  Equation:

$$\min_{\{R_k\}}\sum_{k=0}^{N-1}\sigma_{y_k}^2 2^{-2R_k} \quad \text{s.t.} \quad R = \frac{1}{N}\sum_{k=0}^{N-1}R_k.$$

  Using the Lagrange technique, we first set
  Equation:

$$\frac{\partial}{\partial R_\ell}\left(\sum_k \sigma_{y_k}^2 2^{-2R_k} - \lambda\frac{1}{N}\sum_k R_k\right) = 0 \qquad \forall\ \ell.$$

  Since $2^{-2R_k} \;=\; \left(e^{\ln 2}\right)^{-2R_k} \;=\; e^{-2R_k \ln 2}$, the zero derivative implies
  Equation:

$$0 = -2\ln 2 \cdot 2^{-2R_\ell}\cdot\sigma_{y_\ell}^2 - \frac{\lambda}{N} \Rightarrow R_\ell = -\frac{1}{2}\log_2\left(\frac{\lambda}{-2N\sigma_{y_\ell}^2\ln 2}\right)\forall\ell.$$

  Hence
  Equation:

$$R = \frac{1}{N}\sum_k R_k = -\frac{1}{2N}\sum_k \log_2\left(\frac{\lambda}{-2N\sigma_{y_k}^2\ln 2}\right) = -\frac{1}{2}\log_2\left(\frac{\lambda}{-2N\ln 2}\right) + \frac{1}{2N}\sum_k \log_2\sigma_{y_k}^2$$

  so that
  Equation:

$$-\frac{1}{2}\log_2\left(\frac{\lambda}{-2N\ln 2}\right) \;=\; R-\frac{1}{2}\log_2\left(\prod_k \sigma_{y_k}^2\right)^{1/N}.$$

Rewriting [link] and plugging in the expression above,
Equation:

$$\begin{aligned} R_\ell &= -\frac{1}{2}\log_2\left(\frac{\lambda}{-2N\ln 2}\right)+\frac{1}{2}\log_2\sigma_{y_\ell}^2 \\ &= R-\frac{1}{2}\log_2\left(\prod_k \sigma_{y_k}^2\right)^{1/N}+\frac{1}{2}\log_2\sigma_{y_\ell}^2 \\ &= \boxed{R+\tfrac{1}{2}\log_2\left(\frac{\sigma_{y_\ell}^2}{\left(\prod_{k=0}^{N-1}\sigma_{y_k}^2\right)^{1/N}}\right)}. \end{aligned}$$

- The optimal bitrate allocation expression [link] (lower equation) is meaningful only when $R_\ell \geq 0$, and practical only for integer numbers of quantization levels 2^{-2R_ℓ} (or practical values of R_ℓ for a particular coding scheme). Practical strategies typically
  - set $R_\ell = 0$ to when [link] (lower equation) suggests that the optimal R_ℓ is negative,
  - round positive R_ℓ to practical values, and
  - iteratively re-optimize $\{R_\ell\}$ using these rules until all R_ℓ have practical values.
- Plugging [link] (lower equation) into [link], we find that optimal bit allocation implies
  Equation:

$$\sigma_{q_\ell}^2 \;=\; \gamma_y 2^{-2R}\left(\prod_{k=0}^{N-1}\sigma_{y_k}^2\right)^{1/N} \qquad \forall\ \ell,$$

  which means that, with optimal bit allocation, each coefficient contributes equally to reconstruction error. (Recall a similar property of the Lloyd-Max quantizer.)

Gain over PCM

The total reconstruction error of transform coding with optimal bit allocation is compared to that of uniformly quantized PCM, and the ratio is found to depend on a type of spectral flatness measure.

- With an orthogonal transform and the optimal bit allocation equation 8 from Optimal Bit Allocation (lower equation), the total reconstruction error equals
  Equation:

  We can compare to uniformly quantized PCM, where . Since an orthogonal transform implies
  Equation:

  we have the following gain over PCM:
  Equation:

  Note that the gain is proportional to the ratio between arithmetic and geometric means of the transform coefficient variances. (Note similarities to the spectral flatness measure.) The factor accounts for changes in distribution which affect uniform-quantizer efficiency. For example, if **T** caused uniformly distributed x to become Gaussian distributed y_k, would contribute a 7 dB loss in TC-to-PCM performance. If, on the other hand, x was Gaussian, then y_k would also be Gaussian and .

The Optimal Orthogonal Transform
In this module, we establish that for transform coding, the optimum orthogonal tranform is the Karhunen-Loeve Transform (KLT). Related properties are also discussed.

- Ignoring the effect of transform choice on uniform-quantizer efficiency γ_y, Gain Over PCM suggests that TC reconstruction error can be minimized by choosing the orthogonal transform $\mathbf{T}$ that minimizes the product of coefficient variances. (Recall that orthogonal transforms preserve the arithmetic average of coefficient variances.)

Note:
Eigen-Analysis of Autocorrelation Matrices
Say that $\mathbf{R}$ is the $N \times N$ autocorrelation matrix of a real-valued, wide-sense stationary, discrete time stochastic process. The following properties are often useful:

1. $\mathbf{R}$ is symmetric and Toeplitz. (A symmetric matrix obeys $\mathbf{R} = \mathbf{R}^t$, while a Toeplitz matrix has equal elements on all diagonals.)
2. $\mathbf{R}$ is positive semidefinite or PSD. (PSD means that $\mathbf{x}^t\mathbf{R}\mathbf{x} \geq 0$ for any real-valued $\mathbf{x}$.)
3. $\mathbf{R}$ has an eigen-decomposition
   Equation:
   $$\mathbf{R} = \mathbf{V}\Lambda\mathbf{V}^t,$$
   where $\mathbf{V}$ is an orthogonal matrix ($\mathbf{V}^t\mathbf{V} = \mathbf{I}$) whose columns are eigenvectors $\{\mathbf{v}_i\}$ of $\mathbf{R}$:
   Equation:
   $$\mathbf{V} = (\mathbf{v}_0\mathbf{v}_1 \cdots \mathbf{v}_{N-1}),$$
   and $\mathbf{\Lambda}$ is a diagonal matrix whose elements are the eigenvalues $\{\lambda_i\}$ of $\mathbf{R}$:
   Equation:
   $$\mathbf{\Lambda} = \text{diag}\,(\lambda_0\lambda_1 \cdots \lambda_{N-1}).$$
4. The eigenvectors $\{\lambda_i\}$ of $\mathbf{R}$ are real-valued and non-negative.
5. The product of the eigenvectors equals the determinant ($\prod_{k=0}^{N-1} \lambda_k = |\mathbf{R}|$) and the sum of the eigenvalues equals the trace ($\sum_{k=0}^{N-1} \lambda_k = \sum_k [\mathbf{R}]_{k,k}$).

- *The KLT:* Using the outer product,
  Equation:
  $$\mathbf{y}(m)\mathbf{y}^t(m) = \begin{bmatrix} y_0^2(m) & y_0(m)y_1(m) & \cdots & y_0(m)y_{N-1}(m) \\ y_1(m)y_0(m) & y_1^2(m) & \cdots & y_1(m)y_{N-1}(m) \\ \vdots & \vdots & \ddots & \vdots \\ y_{N-1}(m)y_0(m) & y_{N-1}(m)y_1(m) & \cdots & y_{N-1}^2(m) \end{bmatrix}$$
  Using $[\mathbf{A}]_{k,k}$ to denote the k^{th} diagonal element of a matrix $\mathbf{A}$, matrix theory implies
  Equation:

$$\begin{aligned}
\prod_{k=0}^{N-1}\sigma_{y_k}^2 &= \prod_{k=0}^{N-1}\left[\mathrm{E}\left\{\mathbf{y}\left(m\right)\mathbf{y}^t\left(m\right)\right\}\right]_{k,k} \\
&\geq \mathrm{E}\left\{\mathbf{y}\left(m\right)\mathbf{y}^t\left(m\right)\right\} \\
&= \mathbf{T}\,\mathrm{E}\left\{\mathbf{x}\left(m\right)\mathbf{x}^t\left(m\right)\right\}\mathbf{T}^t \\
&= \underset{\mathbf{I}}{\mathbf{T}^t}\cdot\mathbf{T}\mathrm{E}\left\{\mathbf{x}\left(m\right)\mathbf{x}^t\left(m\right)\right\}\underset{\mathbf{I}}{\mathbf{T}^t}\cdot\mathbf{T} \quad \text{since} \quad \mathbf{T}^t\mathbf{A}| = |\mathbf{A}| = |\mathbf{A}\mathbf{T}| \quad \text{for orthogonal } \mathbf{T} \\
&= \prod_{k=0}^{N-1}\lambda_k(\mathbf{R}_x),
\end{aligned}$$

thus minimization of $\prod_k \sigma_{y_k}^2$ would occur if equality could be established above. Say that the eigen-decomposition of the autocorrelation matrix of $x(n)$, which we now denote by R_x, is
Equation:

$$\mathbf{R}_x = \mathbf{V}_x \Lambda_x \mathbf{V}_x^t$$

for orthogonal eigenvector matrix $\mathbf{V_x}$ and diagonal eigenvalue matrix $\mathbf{\Lambda_x}$. Then choosing $\boxed{\mathbf{T} = \mathbf{V}_x^t}$, otherwise known as the *Karhunen-Loeve transform* (KLT), results in the desired property:
Equation:

$$\mathrm{E}\left\{\mathbf{y}\left(m\right)\mathbf{y}^t\left(m\right)\right\} = \mathrm{E}\left\{\mathbf{V}_x^t\mathbf{x}\left(m\right)\mathbf{x}^t\left(m\right)\mathbf{V}_x\right\} = \mathbf{V}_x^t\mathbf{R}_x\mathbf{V}_x = \mathbf{V}_x^t\mathbf{V}_x\Lambda_x\mathbf{V}_x^t\mathbf{V}_x = \mathbf{\Lambda}_x.$$

To summarize:

1. the orthogonal transformation matrix $\mathbf{T}$ minimizing reconstruction error variance has rows equal to the eigenvectors of the input's $N \times N$ autocorrelation matrix,
2. the variances of the optimal-transform outputs $\left\{\sigma_{y_k}^2\right\}$ are equal to the eigenvalues of the input autocorrelation matrix, and
3. the optimal-transform outputs $\left\{y_0\left(m\right), \cdots, y_{N-1}\left(m\right)\right\}$ are uncorrelated. (Why? Note the zero-valued off-diagonal elements of $\mathbf{R}_y = \mathrm{E}\left\{\mathbf{y}\left(m\right)\mathbf{y}^t\left(m\right)\right\}$.)

- Note that the presence of mutually uncorrelated transform coefficients supports our approach of quantizing each transform output independently of the others.

Example:
2×2 KLT Coder

Recall Example 1 from "Background and Motivation" with Gaussian input having $\mathbf{R}_x = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$. The eigenvalues of $\mathbf{R_x}$ can be determined from the characteristic equation $|\mathbf{R}_x - \lambda\mathbf{I}| = 0$:
Equation:

$$\begin{vmatrix} 1-\lambda & \rho \\ \rho & 1-\lambda \end{vmatrix} = (1-\lambda)^2 - \rho^2 = 0 \quad \Leftrightarrow \quad 1-\lambda = \pm\rho \quad \Leftrightarrow \quad \lambda = 1 \pm \rho.$$

The eigenvector $\mathbf{v_0}$ corresponding to eigenvalue $\lambda_0 = 1+\rho$ solves $\mathbf{R}_x\mathbf{v}_0 = \lambda_0\mathbf{v}_0$. Using the notation $\mathbf{v}_0 = \begin{pmatrix} v_{00} \\ v_{01} \end{pmatrix}$ and $\mathbf{v}_1 = \begin{pmatrix} v_{10} \\ v_{11} \end{pmatrix}$,

Equation:

$$\begin{aligned} v_{00} + \rho v_{01} &= (1+\rho)v_{00} \\ \rho v_{00} + v_{01} &= (1+\rho)v_{01} \end{aligned} \quad \Leftrightarrow \quad v_{01} = v_{00}.$$

Similarly, $\mathbf{R}_x\mathbf{v}_1 = \lambda_1\mathbf{v}_1$ yields

Equation:

$$\begin{aligned} v_{10} + \rho v_{11} &= (1-\rho)v_{10} \\ \rho v_{10} + v_{11} &= (1-\rho)v_{11} \end{aligned} \quad \Leftrightarrow \quad v_{11} = -v_{10}.$$

For orthonormality,

Equation:

$$v_{00}^2 + v_{01}^2 = 1 \quad \Rightarrow \quad \mathbf{v}_0 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Equation:

$$v_{10}^2 + v_{11}^2 = 1 \quad \Rightarrow \quad \mathbf{v}_1 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

Thus the KLT is given by $\mathbf{T} = \mathbf{V}_x^t = (\mathbf{v}_0\ \mathbf{v}_1)^t = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$.

Using the KLT and optimal bit allocation, the error reduction relative to PCM is

Equation:

$$\frac{\sigma_r^2{}_{\text{ TC}}}{\sigma_r^2|_{PCM}} = \frac{\gamma_y}{\gamma_x} \cdot \frac{\sqrt{(1+\rho)(1-\rho)}}{\frac{1}{2}((1+\rho)+(1-\rho))} = \sqrt{1-\rho^2}$$

since $\gamma_y = \gamma_x$ for Gaussian $x(n)$. This value equals 0.6 when $\rho = 0.8$, and 0.98 when $\rho = 0.2$ (compare to Figure 2 from "Background and Motivation").

Performance
Here we analyze the optimal reconstruction error for transform coding. As the number of channels grows to infinity, the performance gain over PCM is shown to depend on the spectral flatness measure. Meanwhile, the performance of transform coding with an infinite number of channels is shown to equal that of DPCM with an infinite-length predictor. However, when the DPCM predictor length is equal to the number of transform coding channels, we show that DPCM always yields better performance.

Asymptotic Performance Analysis

- For an $N \times N$ transform coder, Equation 1 from "Gain over PCM" presented an expression for the reconstruction error variance $\sigma_r^2\big|_{\mathrm{TC}}$ written in terms of the quantizer input variances $\left\{\sigma_{y_k}^2\right\}$. Noting the *N*-dependence on $\sigma_r^2\big|_{TC}$ in Equation 1 from "Gain over PCM" and rewriting it as $\sigma_r^2\big|_{\mathrm{TC},N}$, a reasonable question might be: What is $\sigma_r^2\big|_{\mathrm{TC},N}$ as $N \to \infty$?
- When using the KLT, we know that $\sigma_{y_k}^2 = \lambda_k$ where λ_k denotes the k^{th} eigenvalue of $\mathbf{R}_x$. If we plug these $\sigma_{y_k}^2$ into Equation 1 from "Gain over PCM", we get
  Equation:

$$\sigma_r^2\big|_{\mathrm{TC},N} \;=\; \gamma_y 2^{-2R} \left(\prod_{k=0}^{N-1} \lambda_k \right)^{1/N}.$$

  Writing $\left(\prod_k \lambda_k\right)^{1/N} = \exp\left(\frac{1}{N}\sum_k \ln \lambda_k\right)$ and using the Toeplitz Distribution Theorem (see Grenander & Szego)
  Equation:

$$\text{For any } f(\cdot), \quad \lim_{N\to\infty} \frac{1}{N} \sum_k f\left(\lambda_k\right) \;=\; \frac{1}{2\pi} \int_{-\pi}^{\pi} f\left(S_x\left(e^{j\omega}\right)\right) d\omega$$

  with $f(\cdot) = \ln(\cdot)$, we find that
  Equation:

$$\begin{aligned} \lim_{N\to\infty} \sigma_r^2\Big|_{\mathrm{TC},N} &= \gamma_y 2^{-2R} \exp\left(\frac{1}{2\pi} \int_{-\pi}^{\pi} \ln S_x\left(e^{j\omega}\right) d\omega \right) \\ &= \gamma_y \sigma_x^2 \, 2^{-2R} \, \mathrm{SFM}_x \end{aligned}$$

  where SFM_x denotes the spectral flatness measure of $x(n)$, redefined below for convenience:
  Equation:

$$\mathrm{SFM}_x \;=\; \frac{\exp\left(\frac{1}{2\pi}\int_{-\pi}^{\pi} \ln S_x\left(e^{j\omega}\right) d\omega\right)}{\frac{1}{2\pi}\int_{-\pi}^{\pi} S_x\left(e^{j\omega}\right) d\omega}.$$

  Thus, with optimal transform and optimal bit allocation, asymptotic gain over uniformly quantized PCM is
  Equation:

$$\boxed{G_{\mathrm{TC},N\to\infty} = \frac{\sigma_r^2\big|_{\mathrm{PCM}}}{\sigma_r^2\big|_{\mathrm{TC},N\to\infty}} = \frac{\gamma_x \sigma_x^2 2^{-2R}}{\gamma_y \sigma_x^2 2^{-2R} \mathrm{SFM}_x} = \frac{\gamma_x}{\gamma_y} \mathrm{SFM}_x^{-1}}.$$

- Recall that, for the optimal DPCM system,
  Equation:

$$G_{\text{DPCM},N\to\infty} \;=\; \frac{\sigma_r^2\big|_{\text{PCM}}}{\sigma_r^2\big|_{\text{DPCM},N\to\infty}} \;=\; \frac{\sigma_x^2}{\sigma_e^2\big|_{\min}},$$

  where we assumed that the signal applied to DPCM quantizer is distributed similarly to the signal applied to PCM quantizer and where $\sigma_e^2\big|_{\min}$ denotes the prediction error variance resulting from use of the optimal infinite-length linear predictor:
  Equation:

$$\sigma_e^2|_{\min}=\exp\left(\frac{1}{2\pi}\int_{-\pi}^{\pi}\ln\,S_x\left(e^{j\omega}\right)d\omega\right).$$

  Making this latter assumption for the transform coder (implying $\gamma_y=\gamma_x$) and plugging in $\sigma_e^2\big|_{\min}$ yields the following asymptotic result:
  Equation:

$$\boxed{G_{\text{TC},N\to\infty}=G_{\text{DPCM},N\to\infty}=\text{SFM}_x^{-1}}.$$

  In other words, transform coding with infinite-dimensional optimal transformation and optimal bit allocation performs equivalently to DPCM with infinite-length optimal linear prediction.

Finite-Dimensional Analysis: Comparison to DPCM

- The fact that optimal transform coding performs as well as DPCM in the limiting case does not tell us the relative performance of these methods at practical levels of implementation, e.g., when transform dimension and predictor length are equal and $\ll\infty$. Below we compare the reconstruction error variances of TC and DPCM when the transform dimension *equals* the predictor length. Recalling that
  Equation:

$$G_{\text{DPCM},N-1} \;=\; \frac{\sigma_x^2}{\sigma_e^2|_{\min,N-1}}$$

  and
  Equation:

$$\sigma_e^2|_{\min,N-1}=\frac{|\mathbf{R}_N|}{|\mathbf{R}_{N-1}|}$$

  where $\mathbf{R}_N$ denotes the $N\times N$ autocorrelation matrix of $x(n)$, we find
  Equation:

$$G_{\text{DPCM},N-1}=\sigma_x^2\frac{|\mathbf{R}_{N-1}|}{|\mathbf{R}_N|},\quad G_{\text{DPCM},N-2}=\sigma_x^2\frac{|\mathbf{R}_{N-2}|}{|\mathbf{R}_{N-1}|},\quad G_{\text{DPCM},N-3}=\sigma_x^2\frac{|\mathbf{R}_{N-3}|}{|\mathbf{R}_{N-2}|},\quad\cdots$$

  Recursively applying the equations above, we find

Equation:

$$\prod_{k=1}^{N-1} G_{\mathrm{DPCM},k} \;=\; \left(\sigma_x^2\right)^{N-1} \frac{|\mathbf{R}_1|}{|\mathbf{R}_N|} \;=\; \frac{\left(\sigma_x^2\right)^N}{|\mathbf{R}_N|}$$

which means that we can write
Equation:

$$\left|\mathbf{R}_N| \;=\; \left(\sigma_x^2\right)^N \left(\prod_{k=1}^{N-1} G_{\mathrm{DPCM},k}\right)^{-1}.$$

If in the previously derived TC reconstruction error variance expression
Equation:

$$\sigma_r^2\big|_{\mathrm{TC},N} \;=\; \gamma_y 2^{-2R} \left(\prod_{\ell=0}^{N-1} \lambda_\ell\right)^{1/N}$$

we assume that $\gamma_y = \gamma_x$ and apply the eigenvalue property $\prod_\ell \lambda_\ell = |\mathbf{R}_N|$, the TC gain over PCM becomes
Equation:

$$\begin{aligned} G_{\mathrm{TC},N} \;&=\; \frac{\sigma_r^2\big|_{\mathrm{PCM}}}{\sigma_r^2|_{\mathrm{TC},N}} \;=\; \frac{\gamma_x \sigma_x^2 2^{-2R}}{\gamma_x 2^{-2R} \cdot \sigma_x^2 \left(\prod_{k=1}^{N-1} G_{\mathrm{DPCM},k}\right)^{-1/N}} \\ &=\; \left(\prod_{k=1}^{N-1} G_{\mathrm{DPCM},k}\right)^{1/N} \\ &<\; G_{\mathrm{DPCM},N}. \end{aligned}$$

The strict inequality follows from the fact that $G_{\mathrm{DPCM},k}$ is monotonically increasing with *k*. To summarize, DPCM with optimal length-*N* prediction performs better than TC with optimal $N \times N$ transformation and optimal bit allocation for any finite value of *N*. There is an intuitive explanation for this: the propagation of memory in the DPCM prediction loop makes the *effective* memory of DPCM greater than *N*, while in TC the effective memory is exactly *N*.

Sub-Optimum Orthogonal Transforms

The KLT, which is the optimal orthogonal transform for transform coding, requires knowledge of the source statistics that may be unavailable in practice, and eigendecomposition that may be too expensive in practice. This motivates the consideration of other orthogonal transforms, like the DFT, DCT, and DHT. Here we discuss these and analyze the resulting performance when possible. In addition, we discuss practical matters like real-valued DFTs and fast DCT implementations.

- *Goal:* Recall that the goal of the optimal orthogonal transform was to minimize the ratio of geometric to arithmetic output variances:
  Equation:

$$\frac{\left(\prod_{k=0}^{N-1}\sigma_{y_k}^2\right)^{1/N}}{\frac{1}{N}\sum_{k=0}^{N-1}\sigma_{y_k}^2}.$$

  The ratio [link] attains its maximum value ($=1$) when $\sigma_{y_k}^2$ are equal for all *k* and takes on much smaller values when the difference between the $\sigma_{y_k}^2$ (sometimes called the dynamic range of $\left\{\sigma_{y_k}^2\right\}$) is large.
- *Problem with KLT:* The KLT, i.e., the orthogonal transform minimizing [link], is a function of the input signal statistics. Specifically, the KLT equals the eigenvector matrix of the input autocorrelation matrix. Unfortunately, realistic signals are non-stationary, requiring continual KLT redesign if optimality is to be preserved, and eigenvector computation is computationally intensive, especially for large *N*. Thus, the question becomes: Are there *fixed* orthogonal transforms that do a good job of minimizing the ratio [link] for "typical" input signals? As we will see, the answer is yes...
- *DFT Intuitions:* For the sake of intuition, lets first consider choosing **T** as an orthogonal DFT matrix. In this case, the coefficient variances $\left\{\sigma_{y_k}^2\right\}$ would be samples of the power spectrum and the dynamic range of $\left\{\sigma_{y_k}^2\right\}$ would be determined by the relative input power in different frequency bands. Recalling that asymptotic TC ($N\to\infty$) performance is determined by SFM_x, which has the same geometric-to-arithmetic-average structure as [link]:
  Equation:

$$\mathrm{SFM}_x = \frac{\exp\left(\frac{1}{2\pi}\int_{-\pi}^{\pi}\ln S_x\left(e^{j\omega}\right)d\omega\right)}{\frac{1}{2\pi}\int_{-\pi}^{\pi}S_x\left(e^{j\omega}\right)d\omega} = \lim_{N\to\infty}\left.\frac{\left(\prod_{k=0}^{N-1}S\left(e^{j\omega_k}\right)\right)^{1/N}}{\frac{1}{N}\sum_{k=0}^{N-1}S\left(e^{j\omega_k}\right)}\right|_{\omega_k=2\pi k/N},$$

  we might intuit that the DFT is optimal as $N\to\infty$. The asymptotic optimality of the DFT can, in fact, be proven (see Jayant & Noll). Of course, we don't have much reason to expect that the DFT would be optimal for finite transform dimension *N*. Still, for many signals, it performs well. (See [link].)
- *Other Transforms:* The most commonly used orthogonal transform in speech, image, audio, and video coding is the discrete cosine transform (DCT). The excellent performance of the DCT follows from the fact that it is especially suited to "lowpass" signals, a feature shared by most signals in the previously mentioned applications. Note that there are plenty of signals for which the DCT performs poorly—it just so happens that such signals are not frequently encountered in speech, image, audio, or video. We will describe the DCT and provide intuition regarding it's good "lowpass" performance shortly. Like the DFT, the DCT has fast algorithms which make it extremely practical from an implementation standpoint. Another reasonably popular orthogonal transform, requiring even less in the way of computation, is the discrete Hadamard transform (DHT), also described below. [link] compares DFT, DCT, DHT, and KLT for various transform lengths *N* along with asymptotic TC performance. [link](a) shows gain over PCM when using a lowpass autoregressive (AR) source $\{x(n)\}$ generated from white Gaussian noise $\{v(n)\}$ via:
  Equation:

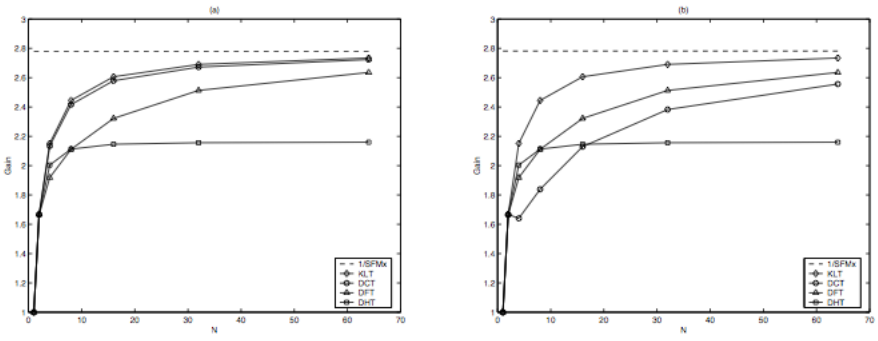
$$X(z) = \frac{1}{1-0.8z^{-1}}V(z),$$

  while [link](b) shows the gain for highpass $\{x(n)\}$:

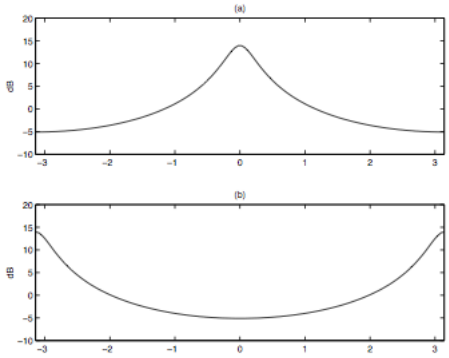
Equation:

$$X(z) = \frac{1}{1+0.8z^{-1}}V(z).$$

See [link] for the power spectra of these two processes.



$G_{\mathrm{TC},N}$ for various transforms and various N on an AR(1) lowpass (left) and highpass (right) sources.



Power spectra of AR(1) sources used in the transform matrix comparisons of [link].

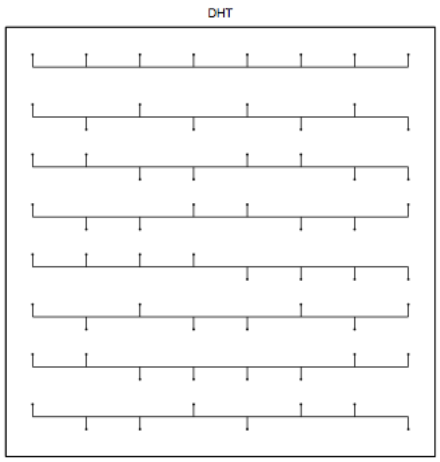
- *The DHT:* The $N \times N$ DHT is defined below for power-of-two N:
Equation:

$$\mathbf{H}_2 = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \qquad \mathbf{H}_{2N} = \frac{1}{\sqrt{2}}\begin{pmatrix}\mathbf{H}_N & \mathbf{H}_N\\ \mathbf{H}_N & -\mathbf{H}_N\end{pmatrix}$$

Note that H_N is orthogonal[footnote], i.e., $\mathbf{H}_N\mathbf{H}_N^t = \mathbf{I}$. As an example
Equation:

$$\mathbf{H}_4 = \frac{1}{2}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\end{pmatrix}$$

[link] illustrates DHT basis vectors for the case $N = 8$. The primary advantage of the DHT is that its implementation can be accomplished very efficiently. [link] suggests that the DHT performs nearly as well as the KLT for $N = 2$ and 4, but its performance falls well short of optimal for larger *N*.



8×8 DHT basis vectors.

Caution: outputs of the Matlab command `hadamard` must be scaled by $1/\sqrt{N}$ to produce *orthogonal* DHT matrices!

- *The DFT:* The normalized[footnote] DFT from $\{x_n\}$ to $\{y_k\}$ is defined below along with its inverse.
Equation:

$$\begin{aligned} y_k &= \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n e^{-j\frac{2\pi}{N}kn}; \quad k = 0 \cdots N-1, \\ x_n &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} y_k e^{j\frac{2\pi}{N}kn}; \quad n = 0 \cdots N-1. \end{aligned}$$

The normalized DFT can be represented by a symmetric unitary[footnote] matrix $\mathbf{W}_N$:
Equation:

$$[\mathbf{W}_N]_{k,n} = \frac{1}{\sqrt{N}} e^{-j\frac{2\pi}{N}kn}; \quad k, n = 0 \cdots N-1.$$

By unitary, we mean that $\mathbf{W}_N \mathbf{W}_N^{*t} = \mathbf{I}$, where $(\cdot)^*$ denotes complex conjugation. Note that a unitary matrix is the complex-valued equivalent of an orthogonal matrix. In practice, the $N \times N$ DFT is implemented using the fast Fourier transform (FFT), which requires $\approx \frac{N}{2} \log_2 N$ complex multiply/adds when *N* is a power of two.
Due to the norm-preserving scale factor $1/\sqrt{N}$, the DFT definitions above differ from those given in most digital signal processing textbooks.
Outputs of the Matlab command `dftmtx` must be scaled by $1/\sqrt{N}$ to produce *unitary* DFT matrices.

- *The Real-Valued DFT:* Since we assume real-valued x_n, complex-valued DFT outputs y_k might seem problematic since transmitting both real and imaginary components would decrease our transmission efficiency. For a real-valued DFT input, however, the DFT outputs exhibit conjugate symmetry which allows us to represent the *N* complex valued outputs with only *N* real-valued numbers. More precisely, real-valued DFT input $\{x_n\}$ implies that DFT output $\{y_k\}$ has the property
Equation:

$$y_k \;=\; \overset{*}{y}_{N-k}, \qquad k = 1, 2, \cdots, N/2,$$

which implies
Equation:

$$\begin{aligned}
\mathrm{Re}\{y_k\} &= \mathrm{Re}\{y_{N-k}\} \qquad k = 1, 2, \cdots, N/2, \\
\mathrm{Im}\{y_k\} &= -\,\mathrm{Im}\{y_{N-k}\} \qquad k = 1, 2, \cdots, N/2, \\
\mathrm{Im}\{y_0\} &= \mathrm{Im}\{y_{N/2}\} \;=\; 0.
\end{aligned}$$

A good method by which to select non-redundant components of the DFT output is:

1. Compute complex-valued $\{y_k\}$ using the standard DFT.
2. Construct real-valued $\{y'_k\}$ from $\{y_k\}$ as follows:
   Equation:

$$\begin{aligned}
y'_0 &= y_0 \;\; (\in \mathbb{R}) \\
y'_1 &= \sqrt{2}\,\mathrm{Im}\,\{y_1\} \\
y'_2 &= \sqrt{2}\,\mathrm{Re}\,\{y_1\} \\
&\vdots \\
y'_{N-3} &= \sqrt{2}\,\mathrm{Im}\,\{y_{N/2-1}\} \\
y'_{N-2} &= \sqrt{2}\,\mathrm{Re}\,\{y_{N/2-1}\} \\
y'_{N-1} &= y_{N/2} \;\; (\in \mathbb{R}).
\end{aligned}$$

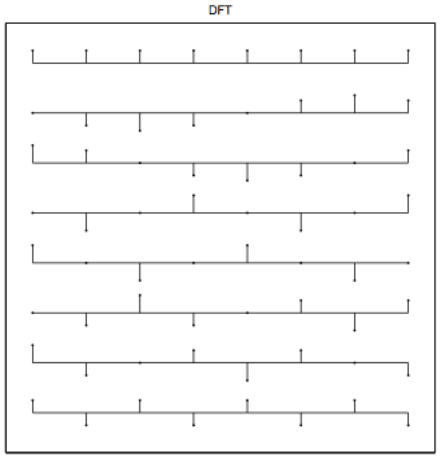
The method above is convenient because (i) it preserves the frequency ordering of the DFT and (ii) it preserves the norm of the DFT output vector, i.e., $\| \mathbf{y}' \| = \| \mathbf{y} \|$. Using the conjugate symmetry property of DFT outputs, we can write the transformation from $\{y_k\}$ from $\{y'_k\}$ as a matrix operation $\mathbf{U}_N$:
Equation:

$$\mathbf{y}' = \underbrace{\begin{pmatrix}
1 & 0 & 0 & 0 & \cdots & & & & \cdots & 0 & 0 & 0 \\
0 & -j/\sqrt{2} & 0 & 0 & \cdots & & & & \cdots & 0 & 0 & j/\sqrt{2} \\
0 & 1/\sqrt{2} & 0 & 0 & \cdots & & & & \cdots & 0 & 0 & 1/\sqrt{2} \\
0 & 0 & -j/\sqrt{2} & 0 & \cdots & & & & \cdots & 0 & j/\sqrt{2} & 0 \\
0 & 0 & 1/\sqrt{2} & 0 & \cdots & & & & \cdots & 0 & 1/\sqrt{2} & 0 \\
\vdots & & & & & & & & & & & \vdots \\
0 & & & \cdots & 0 & -j/\sqrt{2} & 0 & j/\sqrt{2} & 0 & \cdots & & 0 \\
0 & & & \cdots & 0 & 1/\sqrt{2} & 0 & 1/\sqrt{2} & 0 & \cdots & & 0 \\
0 & & & \cdots & 0 & 0 & 1 & 0 & 0 & \cdots & & 0
\end{pmatrix}}_{\mathbf{U}_N} \underbrace{\begin{pmatrix}
\mathrm{Re}\{ \\
\mathrm{Re}\,\{y_1\} + \\
\mathrm{Re}\,\{y_2\} + \\
\vdots \\
\mathrm{Re}\,\{y_{N/2-1}\} + \\
\mathrm{Re}\{y \\
\mathrm{Re}\,\{y_{N/2-1}\} - \\
\vdots \\
\mathrm{Re}\,\{y_2\} - \\
\mathrm{Re}\,\{y_1\} -
\end{pmatrix}}_{\mathbf{y}}$$

The normalization feature guarantees that $\mathbf{U}_N$ is unitary (which is easily checked by inspection). Then $\mathbf{U}_N\mathbf{W}_N$, the product of two unitary matrices, is also unitary. Since $\mathbf{U}_N\mathbf{W}_N$ is actually real-valued (since it takes any real-valued $\mathbf{x}$ to a real-valued $\mathbf{y}$') it should be referred to as orthogonal rather than unitary. Henceforth we rename $\mathbf{U}_N\mathbf{W}_N$ the *real-valued DFT matrix* $\mathbf{W}^{\mathrm{r}}_N$
Equation:

$$\mathbf{W}^{\mathrm{r}}_N \;:=\; \mathbf{U}_N\mathbf{W}_N.$$

It is easily checked that the basis vectors of $\mathbf{W}_N^{\mathrm{r}}$ are sampled sines and cosines at the uniformly spaced frequencies $\{2\pi k/N;\ k = 0, \cdots, N/2\}$. [link] gives an illustration of the real-valued DFT basis vectors for the case $N = 8$.



8×8 Real-valued DFT basis vectors.

Example:
[DFT and Real-valued DFT for $N = 4$]
Equation:

$$\mathbf{W}_4 = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} 3 \\ -1 \\ 4 \\ 2 \end{pmatrix}, \quad \mathbf{W}_4\mathbf{x} = \begin{pmatrix} 4 \\ -1/2 + j3/2 \\ 3 \\ -1/2 - j3/2 \end{pmatrix}.$$

Equation:

$$\mathbf{W}_4^{\mathrm{r}} = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & -\sqrt{2} & 0 & \sqrt{2} \\ \sqrt{2} & 0 & -\sqrt{2} & 0 \\ 1 & -1 & 1 & -1 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} 3 \\ -1 \\ 4 \\ 2 \end{pmatrix}, \quad \mathbf{W}_4^{\mathrm{r}}\mathbf{x} = \begin{pmatrix} 4 \\ 3/\sqrt{2} \\ -1/\sqrt{2} \\ 3 \end{pmatrix}.$$

Recall that when *N* is a power of 2, an *N*-dimensional complex-valued FFT requires $\approx \frac{N}{2} \log_2 N$ complex multiply/adds. When the input is real, however, an *N*-dimensional FFT may be computed using $\approx N \log_2 N$ real multiply/adds (see Sorensen & Jones & Heideman & Burrus TASSP 87).

- *The DCT:* The DCT is defined below along with its inverse

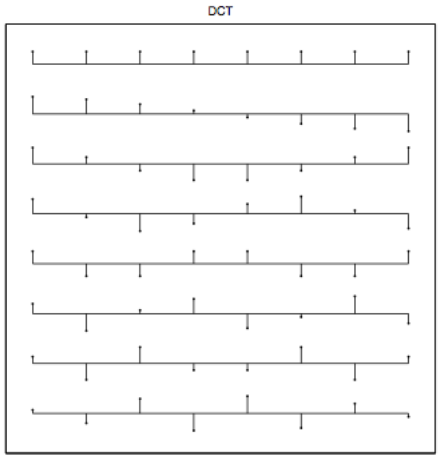
Equation:

$$
\begin{aligned}
y_k &= \sqrt{\frac{2}{N}}\,\alpha_k \sum_{n=0}^{N-1} x_n \cos\frac{(2n+1)k\pi}{2N}; \quad k = 0 \cdots N-1, \\
&\text{for} \quad \alpha_0 = 1/\sqrt{2}, \quad \alpha_{k\neq 0} = 1, \\
x_n &= \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} \alpha_k\, y_k \cos\frac{(2n+1)k\pi}{2N}; \quad n = 0 \cdots N-1,
\end{aligned}
$$

The DCT can be represented by an orthogonal matrix $\mathbf{C}_N$:
Equation:

$$
[\mathbf{C}_N]_{k,n} = \sqrt{\frac{2}{N}}\,\alpha_k \cos\frac{(2n+1)k\pi}{2N}; \quad k, n = 0 \cdots N-1.
$$

See [link] for an illustration of DCT basis vectors when $N = 8$.



8×8 DCT basis vectors.

- *A Fast DCT:* There are a number of fast algorithms to compute the DCT. The method presented below is based on the FFT and leads to intuition concerning the good "lowpass" performance of the DCT.

  1. Create a $2N$-length mirrored version of N-length $\{x_n\}$ (see [link](c)):
     Equation:

$$
\overline{x}_n = \begin{cases} x_n & n = 0, 1, \cdots, N-1, \\ x_{2N-1-n} & n = N, N+1, \cdots, 2N-1. \end{cases}
$$

  2. Compute $\{\overline{y}_k\}$, the $2N$-point DFT of $\{\overline{x}_n\}$:
     Equation:

$$
\overline{y}_k = \frac{1}{\sqrt{2N}} \sum_{n=0}^{2N-1} \overline{x}_n e^{-j\frac{2\pi}{2N}kn} = \sqrt{\frac{2}{N}} e^{j\frac{\pi}{2N}k} \sum_{n=0}^{N-1} x_n \cos\frac{(2n+1)k\pi}{2N}.
$$

  3. Compute $\{y_k\}$, the N-point DCT outputs, from $\{\overline{y}_k\}$:
     Equation:

$$
y_k = e^{-j\frac{\pi}{2N}k} \alpha_k \overline{y}_k; \quad k = 0, 1, \cdots, N-1.
$$

Assuming a real-valued input, the scheme outlined above can be implemented using
Equation:

$$\approx 2N + 2N \log_2 \frac{2N}{2} = 2N\,(1+\log_2 N) \;\; \text{real-valued multiply/adds}$$

where we are assuming use of the real-FFT described previously.

- *DCT vs. DFT Performance for Lowpass Signals:* [link] suggests that DCT and DFT performance both equal KLT performance asymptotically, i.e., as transform dimension $N \to \infty$. Indeed, this can be proven (see Jayant & Noll). A more practical question is: How do DCT and DFT performances compare for finite *N*? To answer this question, we will investigate the effects of input data block length on the DCT and DFT. To start, consider the DFT of an *N*-length input block $\{x_0, \cdots, x_{N-1}\}$:
Equation:

$$y_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n e^{-j\frac{2\pi}{N}kn}; \quad k = 0 \cdots N-1.$$

It can be seen that the DFT outputs $\{X_0, \cdots, X_{N-1}\}$ are samples of the discrete-time Fourier transform (DTFT) at frequencies $\{\omega_k = \frac{2\pi}{N}k;\ k = 0 \cdots N-1\}$:
Equation:

$$X(\omega) \;=\; \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n e^{-j\omega n}, \qquad y_k \;=\; X\big(\tfrac{2\pi}{N}k\big).$$

Now lets consider a periodic extension of $\{x_0, \cdots, x_{N-1}\}$ which repeats this *N*-length sequence a total of *L* times:
Equation:

$$x'_n \;=\; \begin{cases} \frac{1}{\sqrt{L}} x_{\langle n \rangle_N} & -\frac{NL}{2} \le n < \frac{NL}{2} \\ 0 & \text{else.} \end{cases}$$

Above, $\langle n \rangle_N$ denotes "n modulo N" and *L* is assumed even (see [link](a)-(b)). Here is the interesting point: *the DTFT of the NL-length periodic extension equals the DTFT of the original N-length data block when sampled at the frequencies $\{\omega_k\}$!*
Equation:

$$\begin{aligned} X'(\omega_k) &= \frac{1}{\sqrt{NL}} \sum_{n=-NL/2}^{NL/2-1} x'_n e^{-j\frac{2\pi}{N}kn} \\ &= \frac{1}{\sqrt{NL}} \sum_{\ell=-L/2}^{L/2-1} \sum_{m=0}^{N-1} x'_{m+\ell N}\, e^{-j\frac{2\pi}{N}k(m+\ell N)} \\ &= \frac{1}{L\sqrt{N}} \sum_{\ell=-L/2}^{L/2-1} \sum_{m=0}^{N-1} x_m e^{-j\frac{2\pi}{N}k(m+\ell N)} \\ &= \frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} e^{-j\frac{2\pi}{N}km} \\ &= X(\omega_k). \end{aligned}$$

This implies that the overall spectral shape of $X'(\omega)$ will be inherited by the DFT outputs $\{X_0, \cdots, X_{N-1}\}$. So what is the overall shape of $X'(\omega)$? Say that $\{x_n\}$ is a lowpass process. Being lowpass, we expect the time-domain sequence $\{x_n\}$ to look relatively "smooth." If the starting and ending points of the *N*-block, are

different, however, the periodic extension $\{x'_n\}$ will exhibit time-domain discontinuities (see [link](b)) that are uncharacteristic of the process $\{x_n\}$. These discontinuities imply that $X'(\omega)$ will contain high-frequency content not present in the power spectrum of the lowpass input process. Based on our previous findings, if artificial high-frequency content exists at $X'(\omega_k)$, it must also exist at $X(\omega_k) = X_k$. In conclusion, the periodic extension $\{x'_n\}$ provides an intuitive explanation of why short-block DFT analysis of lowpass signals often seems corrupted by "artificial" high-frequency content. So why is this important? Recall that transform performance is proportional to the dynamic range of transform output variances. If the DFT outputs corresponding to otherwise low spectral energy are artificially increased due to short-window effects, the dynamic range of $\{\sigma^2_{y_k}\}$ will decrease, and DFT performance will fall short of optimal.

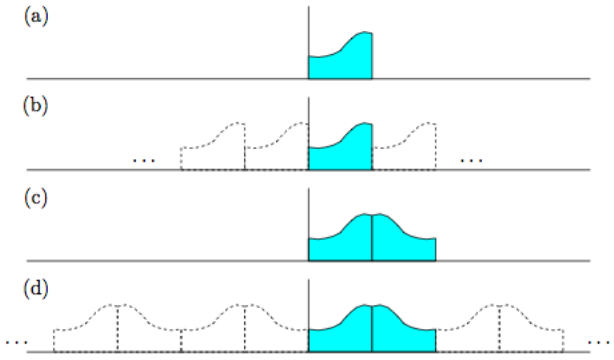


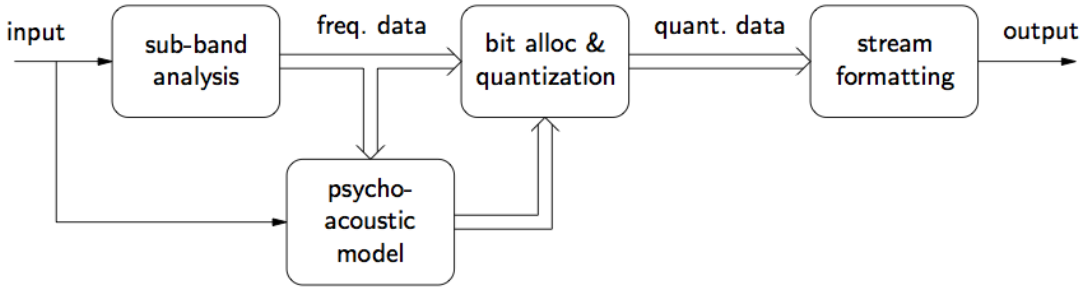
Illustration of periodic extensions inherent to DFT and DCT: (a) *N*-length DFT input block, (b) periodic extension inherent to DFT, (c) equivalent $2N$-length DFT-input-block for DCT, (d) periodic extension inherent to DCT.

Now lets consider the DCT. From derivation of the fast algorithm, we know that the *N* DCT output magnitudes from length-*N* input $\{x_0, \cdots, x_{N-1}\}$ are equal to the first *N* DFT output magnitudes from length-$2N$ input $\{\overline{x}_n\}$—a mirrored version of $\{x_0, \cdots, x_{N-1}\}$. (See [link](c).) Due to the mirroring effect, the periodic extension of $\{\overline{x}_n\}$ will not have the discontinuities present in the periodic extension of $\{x_n\}$, and so a $2N$-point DFT analysis of $\{\overline{x}_n\}$ will not have "artificial" high frequency enhancement. Assuming that the process from which $\{x_n\}$ was extracted is lowpass, the DCT outputs will exhibit large dynamic range, and an improvement over DFT coding performance is expected. This is confirmed by [link](a).

Introduction and Motivation

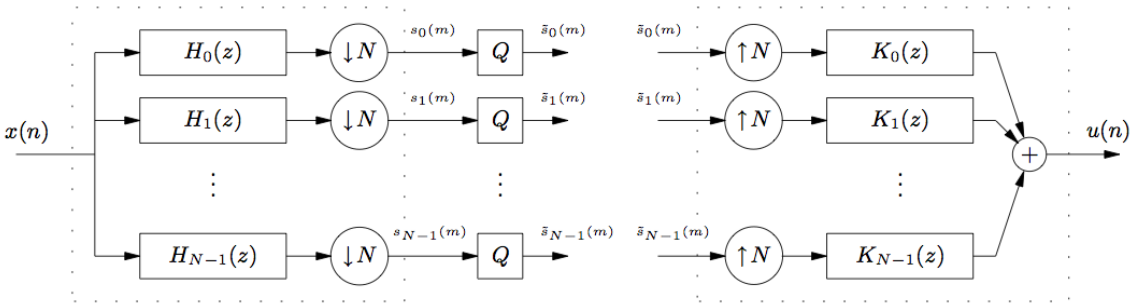
In this module, we give a brief introduction to sub-band coding, its relation to transform coding, and its use in MPEG-style audio coding.

- Sub-band coding is a popular compression tool used in, for example, MPEG-style audio coding schemes (see [link]).



Simplified MPEG-style audio coding system.

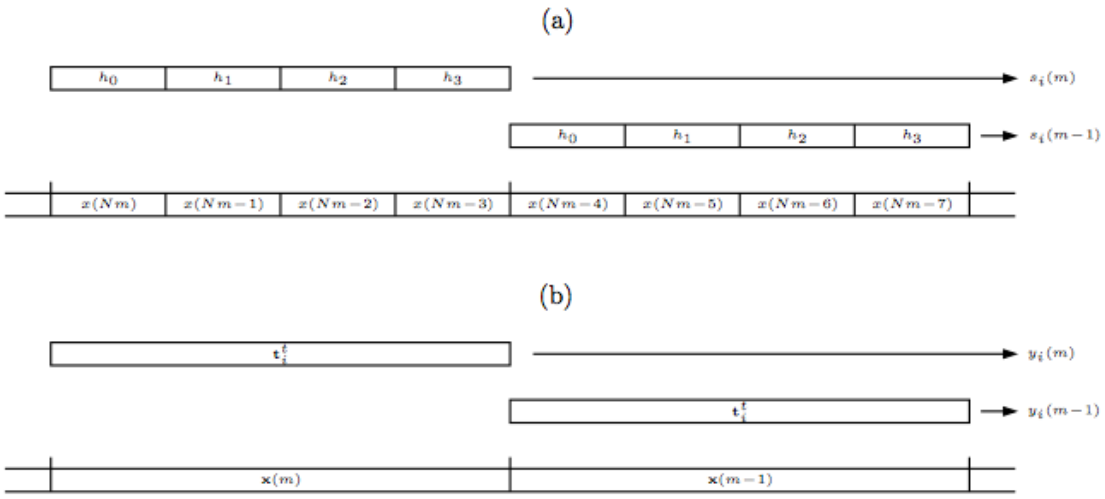
- [link] illustrates a generic subband coder. In short, the input signal is passed through a parallel bank of analysis filters $\{H_i(z)\}$ and the outputs are "downsampled" by a factor of *N*. Downsampling-by-*N* is a process which passes every N^{th} sample and ignores the rest, effectively decreasing the data rate by factor *N*. The downsampled outputs are quantized (using a potentially different number of bits per branch—as in transform coding) for storage or transmission. Downsampling ensures that the number of data samples to store is not any larger than the number of data samples entering the coder; in [link], *N* sub-band outputs are generated for every *N* system inputs.



Sub-band coder/decoder with scalar quantization.

- *Relationship to Transform Coding:* Conceptually, sub-band coding (SC) is very similar to transform coding (TC). Like TC, SC analyzes a block of input data and

produces a set of linearly transformed outputs, now called “subband outputs.” Like TC, these transformed outputs are independently quantized in a way that yields coding gain over straightforward PCM. And like TC, it is possible to derive an optimal bit allocation which minimizes reconstruction error variance for a specified average bit rate. In fact, an *N*-band SC system with length-*N* filters is equivalent to a TC system with $N \times N$ transformation matrix $\mathbf{T}$: the decimated convolution operation which defines the i^{th} analysis branch of [link] is identical to an inner product between an *N*-length input block and $\mathbf{t}_i^t$, the i^{th} row of $\mathbf{T}$. (See [link].)

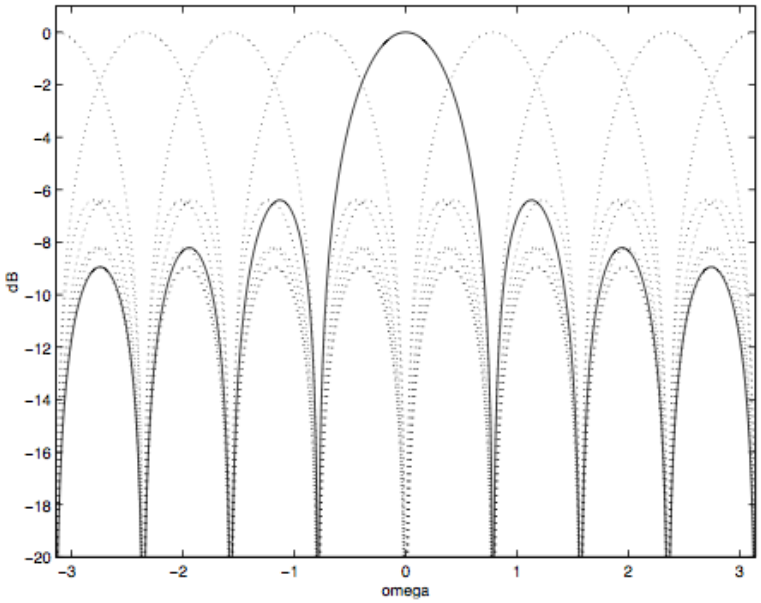


Equivalence between (a) *N*-band sub-band coding with length-*N* filters and (b) $N \times N$ transform coding (shown for $N = 4$). Note: impulse response coefficients $\{h_n\}$ correspond to filter $H_i(z)$.

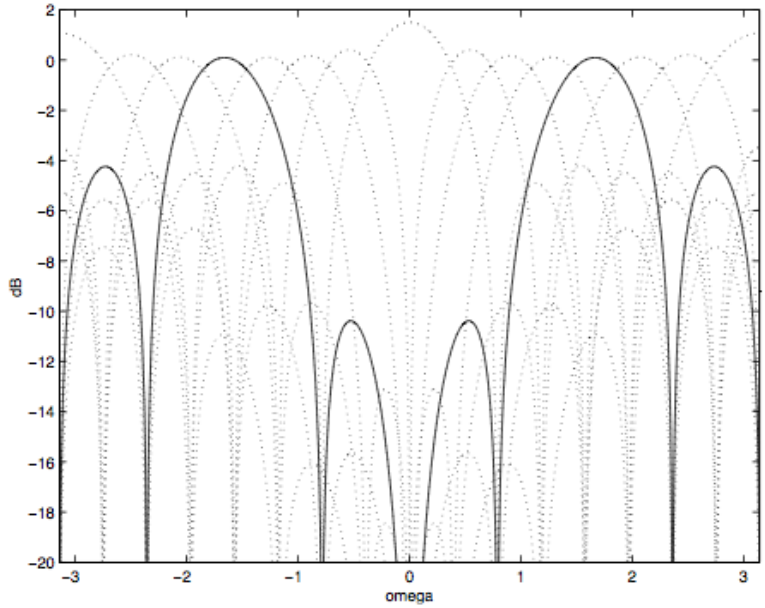
So what kind of frequency responses characterize the most-commonly used transformation matrices? Lets look at the DFT first. For the i^{th} row, we have **Equation:**

$$H_i(\omega)| = \sum_{n=0}^{N-1} e^{-j\frac{2\pi}{N}in} e^{-j\omega n} = \sum_{n=0}^{N-1} e^{-j\ \omega + \frac{2\pi}{N} i\ n} = \frac{\sin\ \frac{N}{2}\ \omega + \frac{2\pi i}{N}}{\sin\ \frac{1}{2}\ \omega + \frac{2\pi i}{N}}.$$

[link] plots these magnitude responses. Note that the i^{th} DFT row acts as a bandpass filter with center frequency $2\pi i/N$ and stopband attenuation of ≈ 6 dB. [link] plots the magnitude responses of DCT filters, where we see that they have even less stopband attenuation.



Magnitude responses of DFT basis vectors for $N = 8$.



Magnitude responses of DCT basis vectors for $N = 8$.

- *Psycho-acoustic Motivations:* We have seen that *N*-band SC with length-*N* filters is equivalent to $N \times N$ transform coding. But is transform coding the best technique to use in high quality audio coders? It turns out that *the key to preserving sonic quality under high levels of compression is to shape the reconstruction error so that the ear will not hear it*. When we talk about psychoacoustics later in the course, we'll see that the properties of noise tolerated by the ear/brain are most easily described in the frequency domain. Hence, *bitrate allocation based on psychoacoustic models is most conveniently performed when SC outputs represent signal components in isolated frequency bands*. In other words, instead of allocating fewer bits to sub-band outputs having a smaller effect on reconstruction error variance, we will allocate fewer bits to sub-band outputs having a smaller contribution to *perceived* reconstruction error. We have seen that length-*N* DFT and DCT filters give a $2\pi/N$ bandwidth with no better than 6 dB of stopband attenuation. The SC filters required for high-quality audio coding require much better stopband performance, say > 90 dB. It turns out that filters with passband width $2\pi/N$, narrow transition bands, and descent stopband attenuation require impulse response lengths $\gg N$. In *N*-band SC there is no constraint on filter length, unlike *N*-band TC. This is the advantage of SC over TC when it comes to audio coding[footnote].
  A similar conclusion resulted from our comparison of DPCM and TC of equal dimension *N*; it was reasoned that the longer "effective" input length of DPCM with *N*-length prediction filtering gave performance improvement relative to TC.
- To summarize, the key differences between transform and sub-band coding are the following.

  1. SC outputs measure relative signal strength in different frequency bands, while TC outputs might not have a strict bandpass correspondence.
  2. The TC input window length is equal to the number of TC outputs, while the SC input window length is usually much greater than number of SC outputs (16$\times$ greater in MPEG).

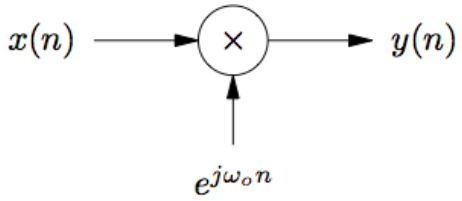
- At first glance SC implementation complexity is a valid concern. Recall that in TC, fast $N \times N$ transforms such as the DCT and DFT could be performed using $\sim N \log_2 N$ multiply/adds! Must we give up this computational efficiency for better frequency resolution? Fortunately the answer is *no*; clever SC implementations are built around fast DFT or DCT transforms and are very efficient as a result. Fast sub-band coding, in fact, lies at the heart of MPEG audio compression (see ISO/IEC 13818-3).

Fundamentals of Multirate Signal Processing

Here we present some background material on multirate signal processing that is necessary to understand the filterbank processing used in sub-band coding. In particular, we describe modulation, upsampling, and downsampling in several domains: the time-domain, z-domain, and DTFT domain. In addition, we describe the aliasing phenomenon.

The presence of upsamplers and downsamplers in the diagram of Figure 2 from "Introduction and Motivation" implies that a basic knowledge of multirate signal processing is indispensible to an understanding of sub-band analysis/synthesis. This section provides the required background.

- *Modulation:*



Modulation using $e^{j\omega_o n}$

[link] illustrates modulation using a complex exponential of frequency ω_o. In the time domain,
Equation:

$$\boxed{y\left(n\right) \;=\; x\left(n\right)e^{j\omega_o n}}.$$

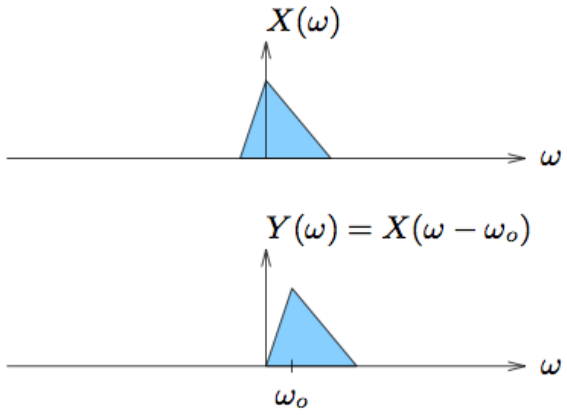
In the z-domain,
Equation:

$$Y\left(z\right) \;=\; \sum_n y\left(n\right)z^{-n} \;=\; \sum_n \left(x\left(n\right)e^{j\omega_o n}\right)z^{-n} \;=\; \sum_n x\left(n\right)\left(e^{-j\omega_o}z\right)^{-n} \;=\; \boxed{X\left(e^{-j\omega_o}z\right)}.$$

We can evaluate the result of modulation in the frequency domain by substituting $z = e^{j\omega}$. This yields
Equation:

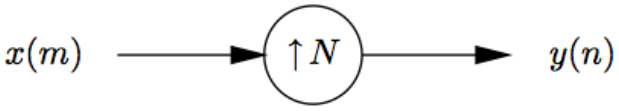
$$Y\left(\omega\right) \;=\; \sum_n y\left(n\right)e^{-j\omega n} \;=\; \boxed{X(\omega - \omega_o)}.$$

Note that $X(\omega - \omega_o)$ represents a shift of $X(\omega)$*up* by ω_o radians, as in [link].



Frequency-domain effect of modulation by $e^{j\omega_o n}$.

- *Upsampling:*



Upsampling by *N*.

[link] illustrates upsampling by factor *N*. In words, upsampling means the insertion of $N-1$ zeros between every sample of the input process. Formally, upsampling can be expressed in the time domain as
Equation:

$$\boxed{y(n) = \begin{cases} x(n/N) & \text{when} n = mN \text{ for} m \in \mathbb{Z} \\ 0 & \text{else.} \end{cases}}$$

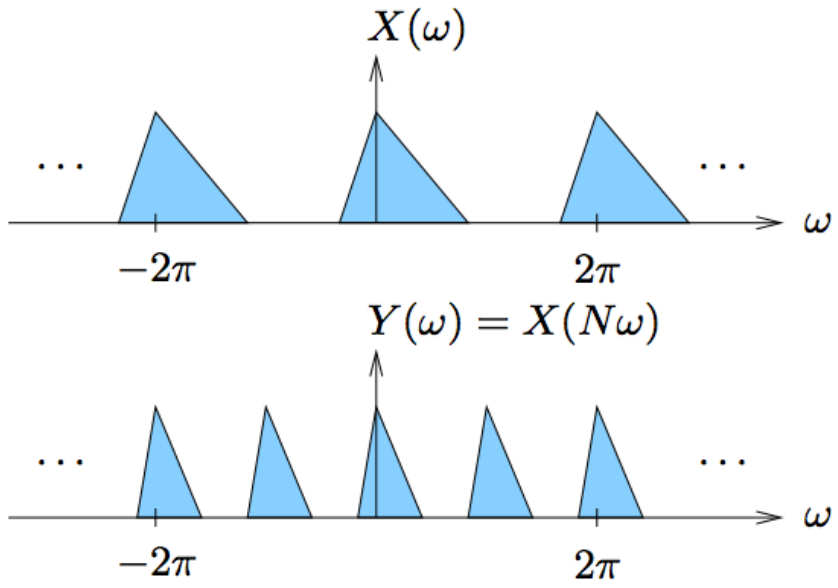
In the *z*-domain, upsampling causes
Equation:

$$Y(z) = \sum_n y(n) z^{-n} = \sum_m x(m) z^{-mN} = \boxed{X\left(z^N\right)},$$

and in the frequency domain,
Equation:

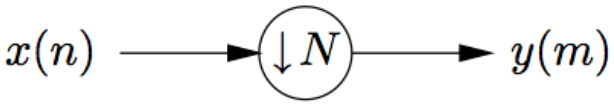
$$Y(\omega) = \sum_n y(n) e^{-j\omega n} = \boxed{X(N\omega)}.$$

As shown in [link], upsampling shrinks $X(\omega)$ by a factor of *N* along the ω axis.



Frequency-domain effects of upsampling by $N = 2$.

- *Downsampling:*



Downsampling by *N*.

[link] illustrates downsampling by factor *N*. In words, the process of downsampling keeps every N^{th} sample and discards the rest. Formally, downsampling can be written as
Equation:

$$\boxed{y(m) \;=\; x(mN).}$$

In the *z* domain,
Equation:

$$Y\left(z\right) \;=\; \sum_{m} y\left(m\right) z^{-m} \;=\; \sum_{m} x\left(mN\right) z^{-m} \;=\; \sum_{n} \widetilde{x}\left(n\right) z^{-n/N},$$

where
Equation:

$$\widetilde{x}\,(n) \;=\; \begin{cases} x(n) & \text{when } n = mN \text{ for } m \in \mathbb{Z} \\ 0 & \text{else.} \end{cases}$$

The neat trick
Equation:

$$\frac{1}{N}\sum_{p=0}^{N-1} e^{j\frac{2\pi}{N}np} \;=\; \begin{cases} 1 & \text{when } n = mN \text{ for } m \in \mathbb{Z} \\ 0 & \text{else} \end{cases}$$

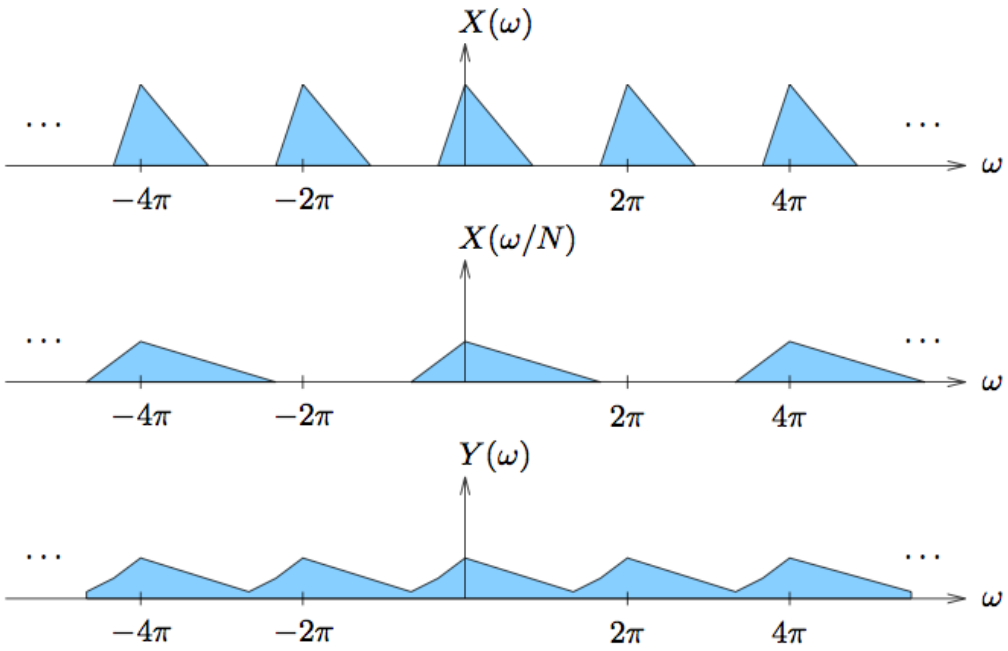
(which is not difficult to prove) allows us to rewrite $\widetilde{x}\,(n)$ in terms of $x(n)$:
Equation:

$$\begin{aligned} Y(z) &= \sum_n x\,(n)\left(\frac{1}{N}\sum_{p=0}^{N-1} e^{j\frac{2\pi}{N}np}\right) z^{-n/N} \\ &= \frac{1}{N}\sum_{p=0}^{N-1}\sum_n x\,(n)\left(e^{-j\frac{2\pi}{N}p} z^{1/N}\right)^{-n} \\ &= \boxed{\tfrac{1}{N}\textstyle\sum_{p=0}^{N-1} X\left(e^{-j\frac{2\pi}{N}p} z^{1/N}\right)}. \end{aligned}$$

Translating to the frequency domain,
Equation:

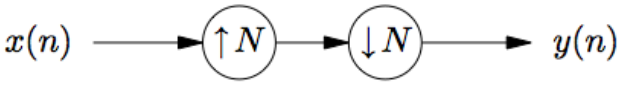
$$\boxed{Y\,(\omega) \;=\; \tfrac{1}{N}\textstyle\sum_{p=0}^{N-1} X\left(\frac{\omega - 2\pi p}{N}\right)}.$$

As shown in [link], downsampling expands each 2π-periodic repetition of $X(\omega)$ by a factor of N along the ω axis. Note the spectral overlap due to downsampling, called “aliasing.”



Frequency-domain effects of downsampling by $N = 2$.

- *Downsample-Upsample Cascade:*



N-Downsampler followed by *N*-upsampler.

Downsampling followed by upsampling (of equal factor *N*) is illustrated by [link]. This structure is useful in understanding analysis/synthesis filterbanks that lie at the heart of sub-band coding schemes. This operation is equivalent to zeroing all but the mN^{th} samples in the input sequence, i.e.,
Equation:

$$y(n) = \begin{cases} x(n) & \text{when} n = mN \text{ for} m \in \mathbb{Z} \\ 0 & \text{else.} \end{cases}$$

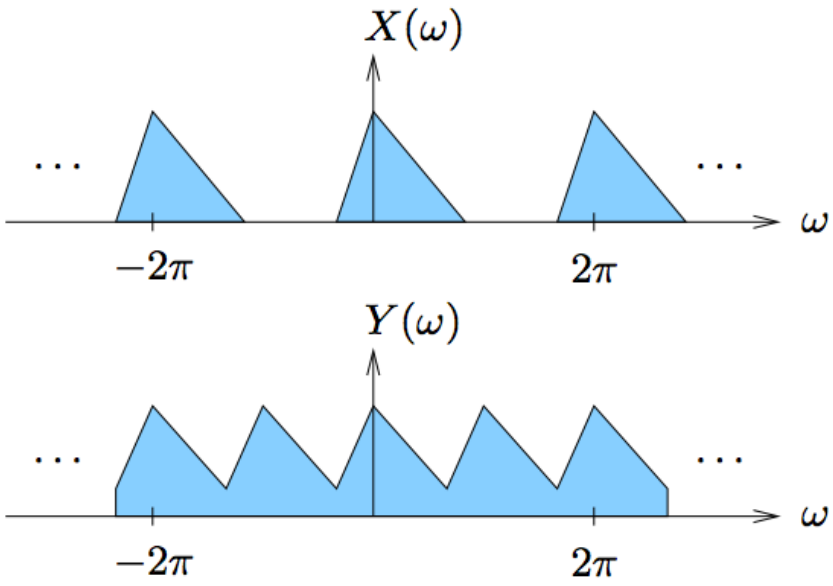
Using trick [link],
Equation:

$$\begin{aligned} Y(z) &= \sum_n y(n) z^{-n} \\ &= \sum_n x(n) \left(\frac{1}{N} \sum_{p=0}^{N-1} e^{j\frac{2\pi}{N}np} \right) z^{-n} \\ &= \frac{1}{N} \sum_{p=0}^{N-1} \sum_n x(n) \left(e^{-j\frac{2\pi}{N}p} z \right)^{-n} \\ &= \boxed{\frac{1}{N} \sum_{p=0}^{N-1} X\left(e^{-j\frac{2\pi}{N}p} z \right)}, \end{aligned}$$

which implies
Equation:

$$\boxed{Y(\omega) = \frac{1}{N} \sum_{p=0}^{N-1} X\left(\omega - \frac{2\pi p}{N} \right)}.$$

The downsampler-upsampler cascade causes the appearance of $2\pi/N$-periodic copies of the baseband spectrum of $X(\omega)$. As illustrated in [link], aliasing may result.

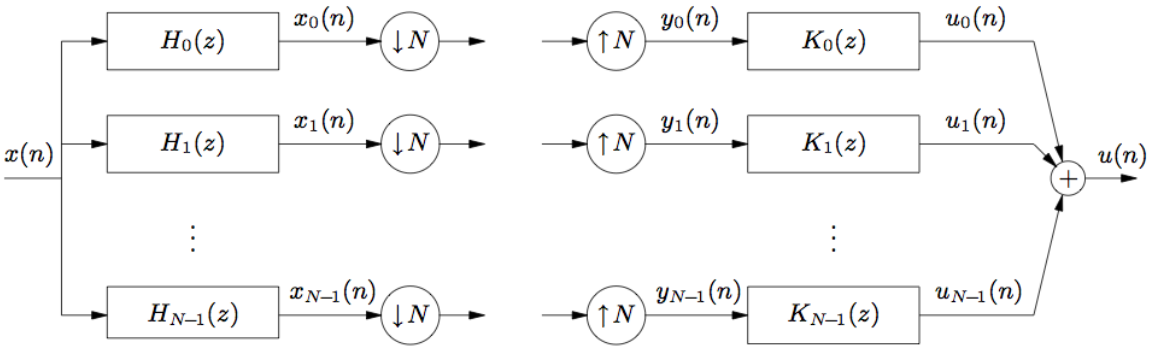


Frequency-domain effects of downsampler-upsampler cascade for $N = 2$.

Uniformly-Modulated Filterbanks

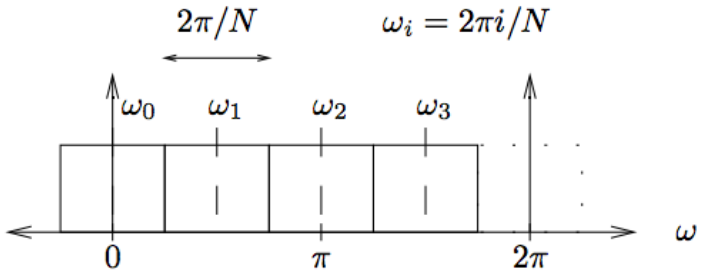
Filterbanks with uniform sub-band bandwidth are described, and the tradeoff between reconstruction error and frequency-selectivity is discussed. The polyphase/DFT filterbank implementaiton is also discussed, along with its computational cost.

- *Perfect Reconstruction Filterbanks:* Recall that in our study of transform coding, we restricted our attention to orthogonal transformation matrices. Orthogonal matrices had the property that, in the absence of quantization error, the reconstruction error was zero. For sub-band coding, "perfect reconstruction" (PR) filterbanks (FBs) are analogous to orthogonal matrices. Specifically, a PR-FB is defined as an analysis/synthesis structure which gives zero reconstruction error when synthesis stage is fed exact (unquantized) copies of analysis outputs. Initially we consider the design of *ideal* sub-band analysis and synthesis FBs and later the design of practical FBs. For the purpose of FB design we ignore the effects of quantization error. Our rational is as follows: the absence of quantization error corresponds to the high bit-rate scenario, in which case we desire that the filtering operations inherent to sub-band coding introduce little or no error of their own. Removing the quantizers from Figure 2 from "Introduction and Motivation", we obtain the analysis/synthesis FBs in [link].



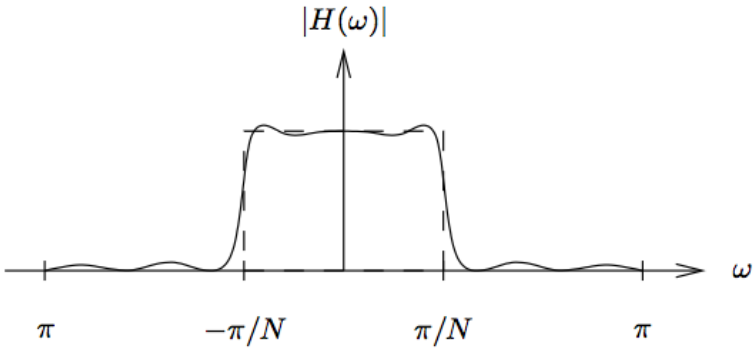
N-band analysis and synthesis filterbanks.

- *Uniform Modulation:* The most conceptually straightforward FB is known as the "uniformly modulated" FB. Uniform modulation means that all branches isolate signal components in non-overlapping frequency bands of equal width $2\pi/N$. We will assume that the i^{th} branch has its frequency band centered at $\omega_i = 2i\pi/N$. (See [link].)



Frequency bands for the uniformly-modulated filterbank ($N = 4$).

- *Analysis FB:* The i^{th} frequency range may be isolated by modulating the input spectrum down by ω_i and lowpass filtering the result. (See the first two stages of the analysis bank in [link].) The ideal lowpass filter has linear phase and magnitude response that is unity for $\omega \in (-\pi/N, \pi/N)$ and zero elsewhere. (See [link].)

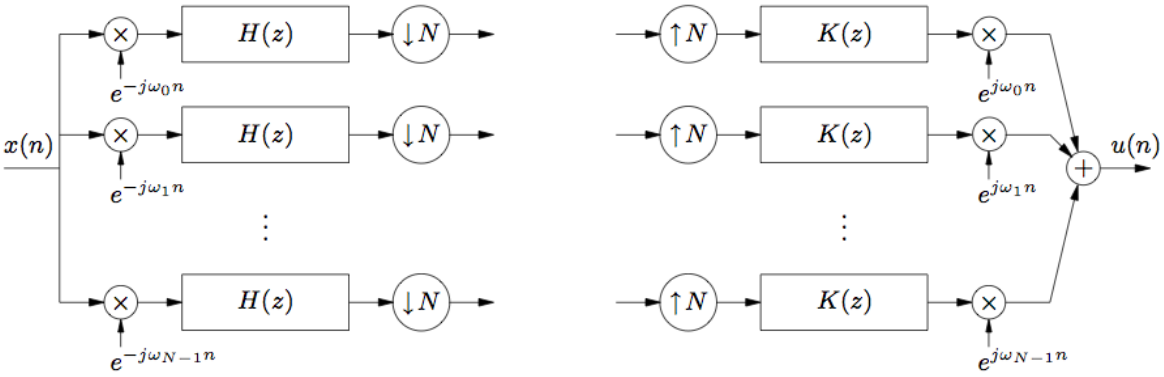


Ideal (dashed) and typical (solid) prototype-filter magnitude responses for the uniformly modulated filterbank.

With ideal lowpass filtering, the resulting signals have (double-sided) bandwidths that are *N* times smaller than the sampling rate. Nyquist's sampling theorem (see Oppenheim & Schafer) says that it is possible to sample signals with this bandwidth at $1/N$ times the filter output rate without loss of information. This sample rate change can implemented via downsampling-by-*N*, resulting in the analysis FB of [link]. Note that the downsampling operation does not induce aliasing when the analysis filter is the ideal lowpass filter described above.

- *Synthesis FB:* To reconstruct the input signal $x(n)$, the synthesis FB must restore the downsampled signals to their original sampling rate, re-modulate them to their original spectral locations, and combine them. Upsampling is the first stage of sampling-rate restoration. Recall from Equation 18 from "Fundamentals of Multirate Signal Processing" (and Figure 8 from "Fundamentals of Multirate Signal Processing") that a downsampler-upsampler cascade creates $N-1$ additional uniformly-spaced spectral copies of the original baseband spectrum. Thus, to remove the unwanted spectral images, an "anti-imaging" lowpass filter is applied to each upsampler's output. Ideally, this lowpass filter is linear phase with magnitude response that is unity for $\omega \in (-\pi/N, \pi/N)$ and zero elsewhere; the same specifications given for the ideal analysis filter. (See [link].) As shown in [link], re-modulation is accomplished by shifting the i^{th} branch *up* by ω_i. When the analysis and synthesis filters share a common phase response, the re-modulator outputs can be combined coherently by a simple summation. Under all of these ideal conditions, the output signal $u(n)$ is a potentially delayed (but otherwise perfect) copy of the input signal $x(n)$:
  Equation:

$$u(n) \;=\; x(n-\delta) \;\; \text{for some delay } \delta.$$



N-band modulated analysis/synthesis filterbanks.

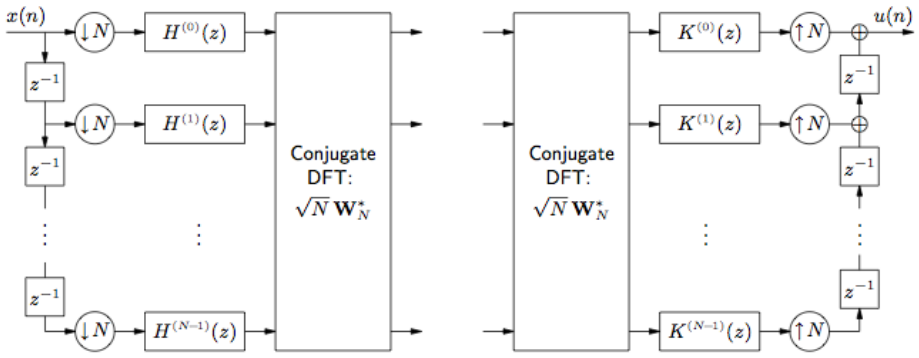
- *Effect of Non-Ideal Filtering:* In practice, the analysis and synthesis filters will not have ideal lowpass responses, and thus the reconstructed output $u(n)$ will not necessarily equal a delayed version of the input $x(n)$. These shortcomings typically result from filter implementations based on a finite number of design parameters. (See [link] for a typical lowpass filter magnitude response.) It should be noted that, under certain conditions, it is possible to design sets of analysis filters $\{H_i(z)\}$ and synthesis filters $\{K_i(z)\}$ with *finite* parameterizations which give the "perfect reconstruction" (PR) property (see Vaidyanathan). Though such filters guarantee PR, they do not act as ideal bandpass filters and thus do not accomplish perfect frequency analysis. (Consider the length-*N* DFT and DCT filter responses: by the orthogonal matrix argument, these are perfectly reconstructing, but from Figure 4 from "Introduction and Motivation" and Figure 5 from "Introduction and Motivation", they are far from perfect bandpass filters!) Due to their limited frequency-selectivity, none of the currently-known PR filterbanks are appropriate for high-quality audio applications. As a result, we focus on the design of filterbanks with

  1. *near*-perfect reconstruction and
  2. good frequency selectivity.

  As we will see, it is possible to design practical filters with excellent frequency selectivity and responses so close to PR that the smallest quantization errors swamp out errors caused by non-PR filtering.
- *Polyphase/DFT Implementation:* When $H(z)$ and $K(z)$ are length-*M* FIR filters, the unique elements in [link] are the *N* uniform-modulation coefficients $\left\{e^{j2\pi n/N}; n = 0, \cdots, N-1\right\}$ and the $2M$ the lowpass filter coefficients $\{h_n\}$ and $\{k_n\}$. It might not be surprising that each half of the uniformly-modulated FB has an implementation that requires only one *N*-dimensional DFT and *M* multiplies to process an *N*-block of input samples. [link] illustrates one such implementation, where the "polyphase" filters $\{H^{(\ell)}(z)\}$ and $\{K^{(\ell)}(z)\}$ are related to the "prototype" filters $H(z)$ and $K(z)$ through the impulse response relations:
  Equation:

$$\begin{aligned} h_m^{(\ell)} &:= h_{mN+\ell} \\ k_m^{(\ell)} &:= k_{mN+\ell} \end{aligned} \quad \text{for} \quad \ell = 0, \cdots, N-1.$$

  The term "polyphase" comes about because the magnitude responses of well-designed $\{H^{(\ell)}(z)\}$ and $\{K^{(\ell)}(z)\}$ are nearly flat, while the slopes of the phase response of these filters differ by small amounts. The equivalence of [link] and [link] will be established in the homework.



Polyphase/DFT implementation of *N*-band uniformly modulated analysis/synthesis filterbanks.

Recognize that the filter computations in [link] occur on downsampled (i.e., low-rate) data, in contrast to those in [link]. To put it another way, all but one of every *N* filter outputs in [link] are thrown away by the downsampler, whereas none of the filter outputs in [link] are thrown away. This reduces the number of required filter computations by a factor of *N*. Additional computational reduction occurs when the DFT is implemented by a fast transform. Below we give a concrete example.

Example:
Computational Savings of Polyphase/FFT Implementation)
Lets take a look at how many multiplications we save by using the polyphase/DFT analysis filterbank in [link] instead of the standard modulated filterbank in [link]. Here we assume that *N* is a power of 2 (see Sorensen, Jones, Heideman & Burrus TASSP 87), so that the DFT can be implemented with a radix-2 FFT. With the standard structure in [link], modulation requires $2N$ real multiplies, and filtering of the complex-valued modulator outputs requires $2 \times N \times M$ additional real multiplies, for each input point $x(n)$. This gives a total of
Equation:

$$\boxed{2N(M+1)} \text{ real multiplies per input.}$$

In the polyphase/FFT structure of [link], it is more convenient to count the number of multiplies required for each block of *N* inputs since each new *N*-block produces one new sample at every filter input and one new *N*-vector at the DFT input. Since the polyphase filters are each length-M/N, filtering the block requires $N \times M/N = M$ real multiplies. Though the standard radix-2 *N*-dimensional complex-valued FFT uses $\frac{N}{2} \log_2 N$ complex multiplies, a real-valued *N*-dimensional FFT can be accomplished in $N \log_2 N$ real multiplies when *N* is a power of 2. This gives a total of
Equation:

$$M + N \log_2 N \text{ real multiplies per } N \text{ inputs, or } \boxed{M/N + \log_2 N} \text{ real multiplies per input!}$$

Say we have $N = 32$ frequency bands and the prototype filter is length $M = 512$ (which turn out to be the values used in the MPEG sub-band filter). Then using the formulas above, the standard implementation requires $\boxed{32832}$ multiplies per input, while the polyphase/DFT implementation requires only $\boxed{21}$!

MPEG Layers 1-3: Cosine-Modulated Filterbanks
Here the "polyphase quadrature" filterbank used in the MPEG audio standards is described in great detail. It has the following practical features: real-valued sub-band outputs, near-perfect reconstruction, and polyphase implementation; and is based on cancellation of adjacent sub-band interference.

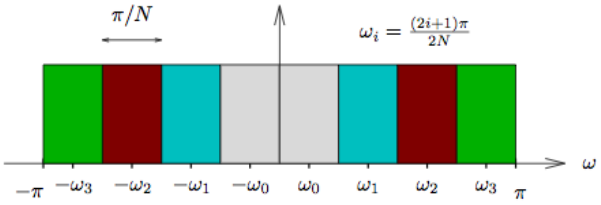
- Though the uniformly modulated filterbank in Figure 4 from "Uniformly-Modulated Filterbanks" was shown to have the fast implementation in Figure 5 from "Uniformly-Modulated Filterbanks", the sub-band outputs are complex-valued for real-valued input, hence inconvenient (at first glance[footnote]) for sub-band coding of real-valued data. In this section we propose a closely related filterbank with the following properties.

  1. Real-valued sub-band outputs (assuming real-valued inputs),
  2. Near-perfect reconstruction,
  3. Polyphase/fast-transform implementation.

  This turns out to be the filterbank specified in the MPEG-1 and 2 (layers 1-3) audio compression standards (see IS0/IEC 13818-3).
  In the structure in Figure 4 from "Uniformly-Modulated Filterbanks", it would be reasonable to replace the standard DFT with a real-valued DFT (defined in the notes on transform coding), requiring $\approx N \log_2 N$ real-multiplies when *N* is a power of 2. Though it is not clear to the author why such a structure was not adopted in the MPEG standards, the cosine modulated filterbank derived in this section has equivalent performance and, with its polyphase/DCT implementation, equivalent implementation cost.

Filter Design

- *Real-valued Sub-band Outputs:* Recall the generic filterbank structure of Figure 1 from "Uniformly-Modulated Filterbanks". For the sub-band outputs to be real-valued (for real-valued input), we require that the impulse responses of $\{H_i(z)\}$ and $\{K_i(z)\}$ are real-valued. We can insure this by allocating the *N* (symmetric) frequency band pairs shown in [link]. The positive and negative halves of each band pair are centered at $\omega_i = \frac{(2i+1)\pi}{2N}$ radians.

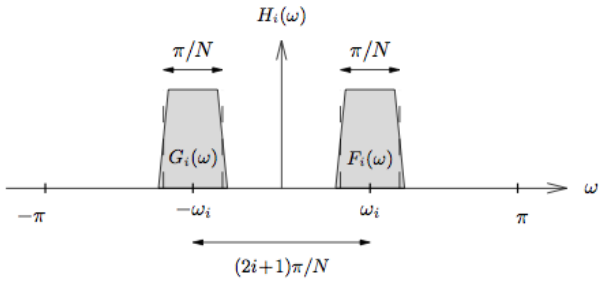


Frequency band pairs for the polyphase quadrature filterbank ($N = 4$).

We can consider each filter $H_i(z)$ as some combination of symmetric positive-frequency and negative-frequency components
Equation:

$$H_i(z) = a_i F_i(z) + b_i G_i(z)$$

as shown in [link].



Positive- and negative-frequency decomposition of $H_i(\omega)$. Note $K_i(\omega)$ will have a similar, if not identical, frequency response.

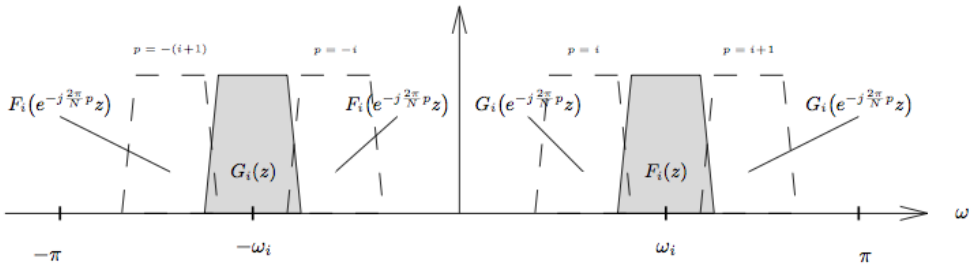
When $b_i = a_i^*$ and the pairs $\{F_i(z), G_i(z)\}$ are modulated versions of the same prototype filter $H(z)$, we can show that $H_i(z)$ must be real-valued:
Equation:

$$\begin{aligned} H_i(z) &= \underbrace{a_i H\left(e^{-j\pi\frac{2i+1}{2N}}z\right)}_{F_i(z)} + \underbrace{a_i^* H\left(e^{j\pi\frac{2i+1}{2N}}z\right)}_{G_i(z)} \\ &= a_i \sum_n h_n e^{j\pi\frac{2i+1}{2N}n} z^{-n} + a_i^* \sum_n h_n e^{-j\pi\frac{2i+1}{2N}n} z^{-n} \\ &= \mathrm{Re}(a_i) \sum_n h_n z^{-n}\left(e^{j\pi\frac{2i+1}{2N}n} + e^{-j\pi\frac{2i+1}{2N}n}\right) + j\,\mathrm{Im}(a_i) \sum_n h_n z^{-n}\left(e^{j\pi\frac{2i+1}{2N}n} - e^{-j\pi\frac{2i+1}{2N}n}\right) \\ &= \mathrm{Re}(a_i) \sum_n h_n z^{-n} \cdot 2\cos\left(\pi\frac{2i+1}{2N}n\right) + j\,\mathrm{Im}(a_i) \sum_n h_n z^{-n} \cdot 2j\sin\left(\pi\frac{2i+1}{2N}n\right) \\ &= 2\sum_n \left[\mathrm{Re}(a_i)\cos\left(\pi\frac{2i+1}{2N}n\right) - \mathrm{Im}(a_i)\sin\left(\pi\frac{2i+1}{2N}n\right)\right] h_n z^{-n} \end{aligned}$$

- *Aliasing Cancellation:* Recall again the generic filterbank in Figure 1 from "Uniformly-Modulated Filterbanks". Here we determine conditions on real-valued $\{H_i(z)\}$ and $\{K_i(z)\}$ which lead to near-perfect reconstruction. It will be insightful to derive an expression for the input to the i^{th} reconstruction filter, $\{y_i(n)\}$. The downsample-upsample-cascade equation Equation 14 from "Fundamentals of Multirate Signal Processing" (fourth equation) implies that
Equation:

$$\begin{aligned} Y_i(z) &= \frac{1}{N}\sum_{p=0}^{N-1} X_i\left(e^{-j\frac{2\pi}{N}p}z\right) \\ &= \frac{1}{N}\sum_{p=0}^{N-1} H_i\left(e^{-j\frac{2\pi}{N}p}z\right) X\left(e^{-j\frac{2\pi}{N}p}z\right) \\ &= \frac{1}{N}\sum_{p=0}^{N-1} \left[a_i F_i\left(e^{-j\frac{2\pi}{N}p}z\right) + a_i^* G_i\left(e^{-j\frac{2\pi}{N}p}z\right)\right] X\left(e^{-j\frac{2\pi}{N}p}z\right) \\ &= \underbrace{\frac{1}{N}\left[a_i F_i(z) + a_i^* G_i(z)\right] X(z)}_{\text{desired}} + \underbrace{\frac{1}{N}\sum_{p=1}^{N-1} \left[a_i F_i\left(e^{-j\frac{2\pi}{N}p}z\right) + a_i^* G_i\left(e^{-j\frac{2\pi}{N}p}z\right)\right] X\left(e^{-j\frac{2\pi}{N}p}z\right)}_{\text{undesired images}}. \end{aligned}$$

Thus the input to the i^{th} reconstruction filter is corrupted by unwanted spectral images, and the reconstruction filter's job is the removal of these images. The reconstruction filter $K_i(z)$ will have a bandpass frequency response similar (or identical) to that of $H_i(z)$ illustrated in [link]. Due to the practical design considerations, neither $K_i(z)$ nor $H_i(z)$ will be perfect bandpass filters, but we will assume that the only significant out-of-band energy passed by these filters will occur in the frequency range just outside of their passbands. (Note the limited "spillover" in [link].) Under these assumptions, the only undesired images in $Y_i(\omega)$ that will not be completely attenuated by $K_i(\omega)$ are the images adjacent to $F_i(\omega)$ and $G_i(\omega)$. Which indices *p* in [link] (third equation) are responsible for these *adjacent* images? [link] (third equation) implies that index $p = \ell$ shifts the frequency response up by $2\pi\ell/N$ radians. Since the passband centers of $F_i(z)$ and $G_i(z)$ are $(2i+1)\pi/N$ radians apart, the passband of $G_i\left(e^{-j\frac{2\pi}{N}p}z\right)$ will reside directly to the left of the passband of $F_i(z)$ when $p = i$. Similarly, the passband of $G_i\left(e^{-j\frac{2\pi}{N}p}z\right)$ will reside directly to the right of the passband of $F_i(z)$ when $p = i+1$. See [link] for an illustration. Using the same reasoning, the passband of $F_i\left(e^{-j\frac{2\pi}{N}p}z\right)$ will reside directly to the right of the passband of $G_i(z)$ when $p = -i$ and directly to the left when $p = -(i+1)$. The only exceptions to this rule occur when $i = 0$, in which case the images to the right of $G_i(z)$ and to the left of $F_i(z)$ are desired, and when $i = N-1$, in which case the images to the left of $G_i(z)$ and to the right of $F_i(z)$ are desired.



Spectral images of $Y_i(\omega)$ not completely attenuated by $K_i(\omega)$.

Based on the arguments above, we can write $\{u_i(n)\}$, the output of the i^{th} reconstruction filter, as follows:
Equation:

$$\begin{aligned} U_i(z) &= K_i(z)Y_i(z) \\ &= \underbrace{\frac{1}{N}K_i(z)\left[a_iF_i(z)X(z) + a_i^*G_i(z)X(z)\right]}_{\text{desired}} \\ &\quad \underbrace{+\frac{1}{N}K_i(z)\left[a_iF_i\left(e^{j\frac{2\pi}{N}i}z\right)X\left(e^{j\frac{2\pi}{N}i}z\right) + a_i^*G_i\left(e^{-j\frac{2\pi}{N}i}z\right)X\left(e^{-j\frac{2\pi}{N}i}z\right)\right]}_{\text{aliasing from inner undesired images when } 1\le i\le N-1} \\ &\quad \underbrace{+\frac{1}{N}K_i(z)\left[a_iF_i\left(e^{j\frac{2\pi}{N}(i+1)}z\right)X\left(e^{j\frac{2\pi}{N}(i+1)}z\right) + a_i^*G_i\left(e^{-j\frac{2\pi}{N}(i+1)}z\right)X\left(e^{-j\frac{2\pi}{N}(i+1)}z\right)\right]}_{\text{aliasing from outer undesired images when } 0\le i\le N-2}. \end{aligned}$$

The previous equation shows that $U_i(z)$ is corrupted by the portions of the undesired images not completely removed by the reconstruction filter $K_i(z)$. In the filterbank context, this undesired behavior is referred to as aliasing. But notice that aliasing contributions to the signal $U(z) = \sum_i U_i(z)$ will vanish if the inner aliasing components in $U_i(z)$ cancel the outer aliasing components in $U_{i-1}(z)$. This happens when
Equation:

$$K_i(z)\left[a_i F_i\left(e^{j\frac{2\pi}{N}i}z\right)X\left(e^{j\frac{2\pi}{N}i}z\right)+a_i^* G_i\left(e^{-j\frac{2\pi}{N}i}z\right)X\left(e^{-j\frac{2\pi}{N}i}z\right)\right]$$
$$=-K_{i-1}(z)\left[a_{i-1}F_{i-1}\left(e^{j\frac{2\pi}{N}i}z\right)X\left(e^{j\frac{2\pi}{N}i}z\right)+a_{i-1}^* G_{i-1}\left(e^{-j\frac{2\pi}{N}i}z\right)X\left(e^{-j\frac{2\pi}{N}i}z\right)\right].$$

which occurs under satisfaction of the two conditions below.
Equation:

$$\begin{aligned} a_i K_i(z)F_i\left(e^{j\frac{2\pi}{N}i}z\right) &= -a_{i-1}K_{i-1}(z)F_{i-1}\left(e^{j\frac{2\pi}{N}i}z\right) \\ a_i^* K_i(z)G_i\left(e^{-j\frac{2\pi}{N}i}z\right) &= -a_{i-1}^* K_{i-1}(z)G_{i-1}\left(e^{-j\frac{2\pi}{N}i}z\right). \end{aligned}$$

We assume from this point on that the real-valued filters $\{H_i(z)\}$ and $\{K_i(z)\}$ are constructed using modulated versions of a lowpass prototype filter $H(z)$. (This assumption is required for the existence of a polyphase filterbank implementation.)
Equation:

$$\begin{aligned} H_i(z) &= a_i F_i(z)+a_i^* G_i(z) \\ K_i(z) &= c_i F_i(z)+c_i^* G_i(z) \end{aligned} \text{ where } \begin{cases} F_i(z) &= H\left(e^{-j\frac{\pi}{2N}(2i+1)}z\right) \\ G_i(z) &= H\left(e^{j\frac{\pi}{2N}(2i+1)}z\right) \end{cases}$$

Then condition [link] (upper equation) becomes
Equation:

$$a_i c_i H\left(e^{-j\frac{\pi}{2N}(2i+1)}z\right)H\left(e^{j\frac{\pi}{2N}(2i-1)}z\right)+a_i c_i^* H\left(e^{j\frac{\pi}{2N}(2i+1)}z\right)H\left(e^{j\frac{\pi}{2N}(2i-1)}z\right)$$
$$=-a_{i-1}c_{i-1}H\left(e^{-j\frac{\pi}{2N}(2i-1)}z\right)H\left(e^{j\frac{\pi}{2N}(2i+1)}z\right)-a_{i-1}c_{i-1}^* H\left(e^{j\frac{\pi}{2N}(2i-1)}z\right)H\left(e^{j\frac{\pi}{2N}(2i+1)}z\right).$$

Lets take a closer look at the products $H\left(e^{-j\frac{\pi}{2N}(2i+1)}z\right)H\left(e^{j\frac{\pi}{2N}(2i-1)}z\right)$ in the previous equation. As illustrated in [link], these products equal zero when $1 \le i \le N/2$ since their passbands do not overlap. Setting these products to zero in [link] (bottom equation) yields the condition
Equation:

$$\boxed{a_i c_i^* = -a_{i-1}c_{i-1}^* \text{for} 1 \le i \le N-1},$$

which can also be shown to satisfy [link] (bottom equation).

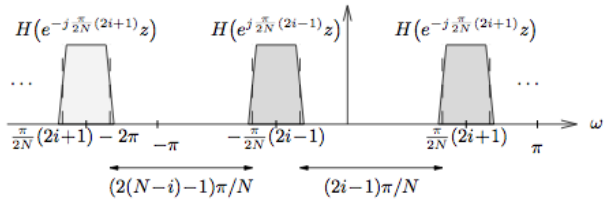


Illustration of vanishing terms in [link] (lower equation).

Next we concern ourselves with the requirements on a_0 and c_0. Assuming [link] is satisfied, we know that inner aliasing in $U_i(z)$ cancels outer aliasing in $U_{i-1}(z)$ for $1 \le i \le N-1$. Hence, from [link] (fourth equation) and [link] (lower equation),
Equation:

$$\begin{aligned} U(z) &= \sum_{i=0}^{N-1} U_i(z) \\ &= \frac{1}{N}\sum_{i=0}^{N-1} K_i(z) H_i(z) X(z) \\ &= \frac{1}{N}\sum_{i=0}^{N-1} \left[c_i H\left(e^{-j\frac{\pi}{2N}(2i+1)}z\right) + c_i^* H\left(e^{j\frac{\pi}{2N}(2i+1)}z\right)\right] \\ &\quad \cdot \left[a_i H\left(e^{-j\frac{\pi}{2N}(2i+1)}z\right) + a_i^* H\left(e^{j\frac{\pi}{2N}(2i+1)}z\right)\right] X(z) \end{aligned}$$

Noting that the passbands of $H\left(e^{-j\frac{\pi}{2N}(2i+1)}z\right)$ and $H\left(e^{j\frac{\pi}{2N}(2i+1)}z\right)$ do not overlap for $1 \le i \le N-2$, we have
Equation:

$$\begin{aligned} U(z) &= \frac{1}{N}\Bigg[\left(a_0 c_0^* + a_0^* c_0\right) H\left(e^{-j\frac{\pi}{2N}}z\right) H\left(e^{j\frac{\pi}{2N}}z\right) \\ &\quad + \left(a_{N-1} c_{N-1}^* + a_{N-1}^* c_{N-1}\right) H\left(e^{-j\frac{\pi}{2N}(2N-1)}z\right) H\left(e^{j\frac{\pi}{2N}(2N-1)}z\right) \\ &\quad + \sum_{i=0}^{N-1}\left(a_i c_i H^2\left(e^{-j\frac{\pi}{2N}(2i+1)}z\right) + a_i^* c_i^* H^2\left(e^{j\frac{\pi}{2N}(2i+1)}z\right)\right)\Bigg] X(z). \end{aligned}$$

The first two terms in [link] (third equation) represent aliasing components that prevent flat overall response at $\omega = 0$ and $\omega = \pi$, respectively. These aliasing terms vanish when
Equation:

$$\boxed{\begin{aligned} a_0 c_0^* &= -a_0^* c_0 \\ a_{N-1} c_{N-1}^* &= -a_{N-1}^* c_{N-1} \end{aligned}}$$

What remains is
Equation:

$$U(z) = \frac{1}{N}\sum_{i=0}^{N-1}\left(a_i c_i H^2\left(e^{-j\frac{\pi}{2N}(2i+1)}z\right) + a_i^* c_i^* H^2\left(e^{j\frac{\pi}{2N}(2i+1)}z\right)\right) X(z).$$

- *Phase Distortion:* Perfect reconstruction requires that the analysis/synthesis system has no phase distortion. To guarantee the absence of phase distortion, we require that the composite system
Equation:

$$Q(z) := \frac{U(z)}{X(z)} = \frac{1}{N}\sum_{i=0}^{N-1} a_i c_i H^2\left(e^{-j\frac{\pi}{2N}(2i+1)}z\right) + a_i^* c_i^* H^2\left(e^{j\frac{\pi}{2N}(2i+1)}z\right)$$

has a linear phase response. (Recall that a linear phase response is equivalent to a pure delay in the time domain.) This linear-phase constraint will provide the final condition used to specify the constants $\{a_i\}$ and $\{c_i\}$. We start by examining the impulse response of $Q(z)$. Using a technique analogous to [link] (fifth equation), we can write
Equation:

$$Q(z) = \frac{2}{N}\sum_{n=0}^{2M-2}\left(\sum_{i=0}^{N-1} \mathrm{Re}(a_i c_i)\cos\left(\pi\frac{2i+1}{2N}n\right) - \mathrm{Im}(a_i c_i)\sin\left(\pi\frac{2i+1}{2N}n\right)\right)\left(\sum_k h_k h_{n-k}\right) z^{-n}$$

Above, we have used the property that multiplication in the z-domain implies convolution in the time domain. For $Q(z)$ to be linear phase, it's impulse response must be symmetric. Let us assume that the prototype filter $H(z)$ is linear phase, so that $\{h_n\}$ is symmetric. Thus $\sum_k h_m h_{n-k}$ is symmetric about $n = M-1$, and thus for linear phase $Q(z)$, we require that the quantity
Equation:

$$\sum_{i=0}^{N-1} \mathrm{Re}\left(a_i c_i\right) \cos\left(\pi \frac{2i+1}{2N} n\right) - \mathrm{Im}\left(a_i c_i\right) \sin\left(\pi \frac{2i+1}{2N} n\right)$$

is symmetric about $n = M-1$, i.e.,
Equation:

$$\begin{aligned} &\sum_{i=0}^{N-1} \mathrm{Re}\left(a_i c_i\right) \cos\left(\pi \frac{2i+1}{2N} (M-1+n)\right) - \mathrm{Im}\left(a_i c_i\right) \sin\left(\pi \frac{2i+1}{2N} (M-1+n)\right) \\ = &\sum_{i=0}^{N-1} \mathrm{Re}\left(a_i c_i\right) \cos\left(\pi \frac{2i+1}{2N} (M-1-n)\right) - \mathrm{Im}\left(a_i c_i\right) \sin\left(\pi \frac{2i+1}{2N} (M-1-n)\right) \end{aligned}$$

for $n = 0, \cdots, M-1$. Using trigonometric identities, it can be shown that the condition above is equivalent to
Equation:

$$0 \;=\; \sum_{i=0}^{N-1} \sin\left(\pi \frac{2i+1}{2N} n\right) \left[\mathrm{Re}\left(a_i c_i\right) \sin\left(\pi \frac{2i+1}{2N} (M-1)\right) + \mathrm{Im}\left(a_i c_i\right) \cos\left(\pi \frac{2i+1}{2N} (M-1)\right)\right],$$

which is satisfied when
Equation:

$$\frac{\mathrm{Im}(a_i c_i)}{\mathrm{Re}(a_i c_i)} \;=\; -\frac{\sin\left(\pi \frac{2i+1}{2N}(M-1)\right)}{\cos\left(\pi \frac{2i+1}{2N}(M-1)\right)} \;= \tan\left(-\pi \frac{2i+1}{2N}(M-1)\right).$$

Restricting $|a_i| = |c_i| = 1$, the previous equation requires that
Equation:

$$\boxed{a_i c_i \;=\; e^{-j\pi \frac{2i+1}{2N}(M-1)}}.$$

It can be easily verified that the following $\{a_i\}$ and $\{c_i\}$ satisfy conditions [link], [link], and [link]:
Equation:

$$\boxed{\begin{aligned} a_i &= e^{-j\pi \frac{M+N-1}{4N}(2i+1)} \\ c_i &= e^{-j\pi \frac{M-N-1}{4N}(2i+1)}. \end{aligned}}$$

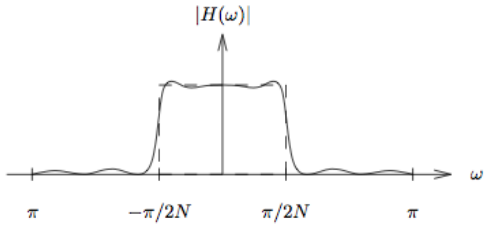
Plugging these into the expression for $H_i\left(z\right)$ we find that
Equation:

$$
\begin{aligned}
H_i(z) &= a_i H\left(e^{-j\pi\frac{2i+1}{2N}}z\right) + a_i^* H\left(e^{j\pi\frac{2i+1}{2N}}z\right) \\
&= \sum_{n=0}^{M-1}\left(a_i e^{j\pi\frac{2i+1}{2N}n} + a_i^* e^{-j\pi\frac{2i+1}{2N}n}\right)h_n z^{-n} \\
&= \sum_{n=0}^{M-1}\left(e^{j\pi\frac{2i+1}{2N}\left(n-\frac{M+N-1}{2}\right)} + e^{-j\pi\frac{2i+1}{2N}\left(n-\frac{M+N-1}{2}\right)}\right)h_n z^{-n} \\
&= \sum_{n=0}^{M-1} \underbrace{\boxed{2\cos\left(\pi\tfrac{2i+1}{2N}\left(n-\tfrac{M+N-1}{2}\right)\right)h_n}}_{\text{impulse response of } H_i(z)} z^{-n}.
\end{aligned}
$$

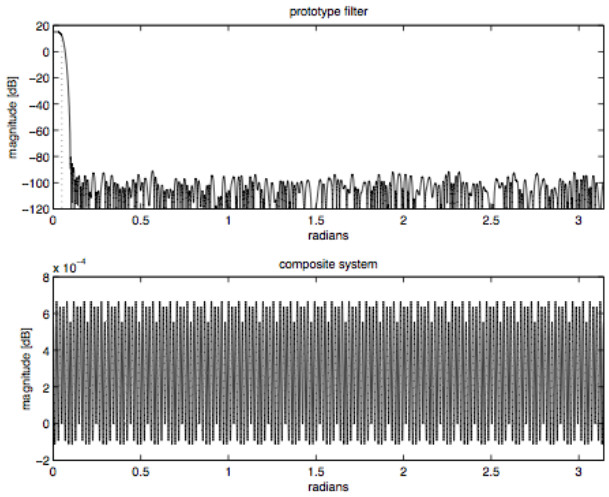
Repeating this procedure for $K_i(z)$ yields
Equation:

$$
K_i(z) = \sum_{n=0}^{M-1} \underbrace{\boxed{2\cos\left(\pi\tfrac{2i+1}{2N}\left(n-\tfrac{M-N-1}{2}\right)\right)h_n}}_{\text{impulse response of } K_i(z)} z^{-n}.
$$

At this point we make a few comments on the design of the lowpass prototype $H(z)$. The perfect $H(z)$ would be an ideal linear-phase lowpass filter with cutoff at $\omega = \pi/2N$, as illustrated in [link]. Such a filter would perfectly separate the subbands as well as yield flat composite magnitude response, as per [link]. Unfortunately, however, this perfect filter is not realizable with a finite number of filter coefficients. So, what we really want is a finite-length FIR filter having good frequency selectivity, nearly-flat composite response, and linear phase. The length-512 prototype filter specified in the MPEG standards is such a filter, as evidenced by the responses in [link]. Unfortunately, the standards do not describe how this filter was designed, and a thorough discussion of multirate filter design is outside the scope of this course. For more on prototype filter design, we point the interested reader to page 358 of Vaidyanathan or Crochiere & Rabiner.



Ideal (dashed) and typical (solid) prototype-filter magnitude responses for the cosine-modulated filterbank. Note bandwidth relative to [link].



Magnitude response of $|H(\omega)|$ of MPEG prototype filter and the resulting composite response $|Q(\omega)|$, where $N = 32$ and $M = 16N = 512$.

To conclude, [link] (fourth equation) and [link] give impulse response expressions for a set of real-valued filters that comprise a near-perfectly reconstructing filterbank (under suitable selection of $\{h_i\}$). This is commonly referred to[footnote] as a “cosine-modulated filterbank” because all filters are based on cosine modulations of a real-valued linear-phase lowpass prototype $H(z)$. The near-perfect reconstruction property follows from the frequency-domain cancellation of adjacent-spectrum aliasing and the lack of phase distortion.
It should be noted that our derivation of the cosine modulated filterbank is similar to that in Rothweiler ICASSP 83 except for the treatments of phase distortion. See Chapter 8 of Vaidyanathan for a more comprehensive view of cosine-modulated filterbanks.
The MPEG standards refer to this filterbank as a “polyphase quadrature” filterbank (PQF), the name given to the technique by an early technical paper: Rothweiler ICASSP 83

- *Polyphase Implementations:* Recall the uniformly modulated filterbank in Figure 4 from "Uniformly-Modulated Filterbanks", whose combined modulator-filter coefficients can be constructed using products of the terms h_n and $e^{j\frac{\pi}{N}in}$. Figure 5 from "Uniformly-Modulated Filterbanks" shows a computationally-efficient polyphase/DFT implementation of the analysis filter which requires only *M* multiplies and one *N*-dimensional DFT computation for calculation of *N* subband outputs. We might wonder: Is there a similar polyphase/fast-transform implementation of the cosine-modulated filterbank derived in this section? From [link] (fourth equation), we see that the impulse responses of $\{H_i(z)\}$ are products of the terms h_n and $\cos\left(\pi\frac{2i+1}{2N}\left(n-\frac{M+N-1}{2}\right)\right)$ for $n = 0, \cdots, M-1$. Note that the inverse-DCT matrix ${C_n}^t$ can be specified via components with form similar to the cosine term in [link] (fourth equation):
Equation:

$$\left[\mathbf{C}_N^t\right]_{i,n} = \sqrt{\frac{2}{N}}\,\alpha_n \cos\left(\pi\frac{(2i+1)}{2N}n\right); \quad i,n = 0\cdots N-1.$$
$$\text{for} \quad \alpha_0 = 1/\sqrt{2}, \quad \alpha_{n\neq 0} = 1.$$

Thus it may not be surprising that there exist polyphase/DCT implementations of the cosine-modulated filterbank. Indeed, one such implementation is specified in the MPEG-2 audio compression standard (see ISO/IEC 13818-3). This particular implementation is the focus of the next section.

MPEG Filterbank Implementation

- Since MPEG audio compression standards are so well-known and widespread, a detailed look at the MPEG filterbank implementation is warranted. The cosine-modulated, or polyphase-quadrature filterbank described in the previous section is used in MPEG Layers 1-3. (The MPEG hierarchy will be described in a later chapter.) This section discusses the specific implementation suggested by the MPEG-2 standard (see ISO/IEC 13818-3).
- The MPEG standard specifies 512 prototype filter coefficients, the first of which is zero. To adapt the MPEG filter to our cosine-modulated-filterbank framework, we append a zero-valued 513$^{\text{th}}$ coefficient so that the resulting MPEG prototype filter becomes symmetric and hence linear phase. Since the standard specifies $N = 32$ frequency bands, we have
  Equation:

$$M \;=\; 513 \;=\; 16N + 1.$$

  Plugging this value of *M* into the filter expressions [link] (fourth equation) and [link], the 2π-periodicity of the cosine implies that they may be rewritten as follows.
  Equation:

$$\begin{aligned} H_i(z) &= \sum_{n=0}^{16N-1} \underbrace{2\cos\left(\pi\tfrac{2i+1}{2N}\left(n-\tfrac{N}{2}\right)\right)h_n}_{\text{impulse response of } H_i(z)} z^{-n} \\ K_i(z) &= \sum_{n=0}^{16N-1} \underbrace{2\cos\left(\pi\tfrac{2i+1}{2N}\left(n+\tfrac{N}{2}\right)\right)h_n}_{\text{impulse response of } K_i(z)} z^{-n}. \end{aligned}$$

- *Encoding:* Here we derive the encoder filterbank implementation suggested in the MPEG-2 standard (see ISO/IEC 13818-3). Using $x_i(n)$ to denote the output of the i^{th} analysis filter, we have
  Equation:

$$x_i(n) \;=\; \sum_{k=0}^{16N-1} \left[2\cos\left(\pi\tfrac{2i+1}{2N}\left(k-\tfrac{N}{2}\right)\right)h_k\right] x(n-k).$$

  The relationship between $x_i(n)$ and its downsampled version $s_i(m)$ is given by
  Equation:

$$s_i(m) = x_i(mN),$$

  so that the downsampled analysis output $s_i(m)$ can be written as
  Equation:

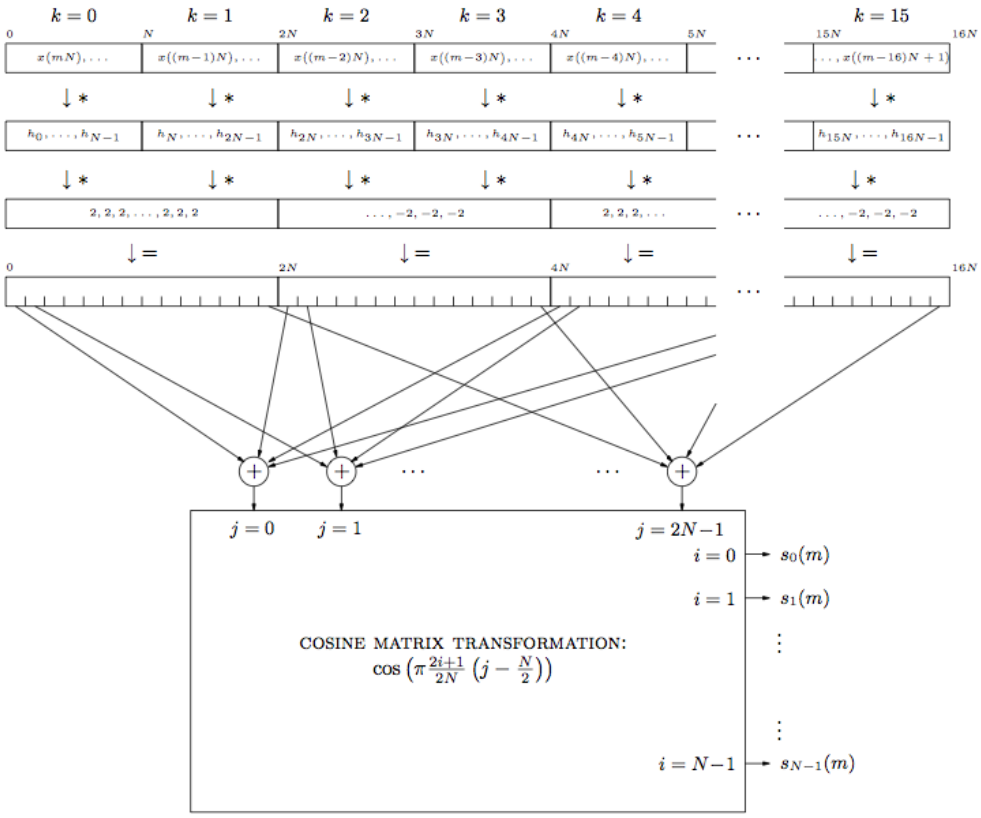
$$s_i(m) \;=\; \sum_{n=0}^{16N-1} \left[2\cos\left(\pi\tfrac{2i+1}{2N}\left(n-\tfrac{N}{2}\right)\right)h_n\right] x(mN-n).$$

  Using the substitution $n = kN + \ell$ for $0 \le \ell \le N-1$,
  Equation:

$$s_i(m) = 2\sum_{k=0}^{15}\sum_{\ell=0}^{N-1} \underbrace{\cos\left(\pi\tfrac{2i+1}{2N}\left(kN+\ell-\tfrac{N}{2}\right)\right)}_{\substack{\text{repeats every 4 increments of } k \\ \text{sign changes every 2 increments of } k}} h_{kN+\ell}\, x((m-k)N-\ell)$$

$$= \sum_{k=0}^{15}\sum_{\ell=0}^{N-1} \underbrace{\cos\left(\pi\tfrac{2i+1}{2N}\left(\langle k\rangle_2 N+\ell-\tfrac{N}{2}\right)\right)}_{\text{repeats every 2 increments of } k} \underbrace{2(-1)^{\lfloor k/2\rfloor} h_{kN+\ell}}_{\text{analysis window}}\, x((m-k)N-\ell)$$

[link] illustrates this process.



MPEG encoder filterbank implementation suggested in ISO/IEC 13818-3.

- *Decoding:* Here we derive the dencoder filterbank implementation suggested in the MPEG-2 standard (see ISO/IEC 13818-3). Using $y_i(n)$ to denote the output of the i^{th} upsampler,
  Equation:

$$u_i(n) = \sum_{k=0}^{16N-1}\left[2\cos\left(\pi\tfrac{2i+1}{2N}\left(k+\tfrac{N}{2}\right)\right)h_k\right] y_i(n-k).$$

The input to the upsampler $s_i(m)$ is related to the output $y_i(n)$ by
Equation:

$$y_i(n) = \begin{cases} s_i(n/N) & \text{when } n/N \in \mathbb{Z} \\ 0 & \text{else,} \end{cases}$$

so that
Equation:

$$u_i(n) \;=\; \textstyle\sum_{\{k:\,\frac{n-k}{N}\in\mathbb{Z}\}}\left[2\cos\left(\pi\frac{2i+1}{2N}\left(k+\frac{N}{2}\right)\right)h_k\right]s_i\left(\frac{n-k}{N}\right).$$

Lets write $n = mN+\ell$ for $0\le\ell\le N-1$ and $k = pN+q$ for $0\le q\le N-1$. Then due to the restricted ranges of ℓ and q,
Equation:

$$\frac{n-k}{N} \;=\; m-p+\frac{\ell-q}{N} \;\in\; \mathbb{Z} \quad\Rightarrow\quad \ell \;=\; q.$$

Using these substitutions in the previous equation for $u_i(n)$,
Equation:

$$u_i(mN+\ell) \;=\; 2\textstyle\sum_{p=0}^{15}\cos\left(\pi\frac{2i+1}{2N}\left(pN+\ell+\frac{N}{2}\right)\right)h_{pN+\ell}\,s_i(m-p).$$

Summing $u_i(mN+\ell)$ over *i* to create $u(mN+\ell)$,
Equation:

$$\begin{aligned} u(mN+\ell) \;&=\; 2\sum_{i=0}^{N-1}\sum_{p=0}^{15}\underbrace{\cos\left(\pi\frac{2i+1}{2N}\left(pN+\ell+\frac{N}{2}\right)\right)}_{\substack{\text{repeats every 4 increments of } p\\ \text{sign changes every 2 increments of } p}}h_{pN+\ell}\,s_i(m-p) \\ &=\; \sum_{p=0}^{15}\underbrace{2(-1)^{\lfloor p/2\rfloor}h_{pN+\ell}}_{\text{synthesis window}}\;\underbrace{\sum_{i=0}^{N-1}\cos\left(\pi\frac{2i+1}{2N}\left(\langle p\rangle_2 N+\ell+\frac{N}{2}\right)\right)}_{=\begin{cases}\cos\left(\pi\frac{2i+1}{2N}\left(\ell+\frac{N}{2}\right)\right) & p\text{ even}\\ \cos\left(\pi\frac{2i+1}{2N}\left(\ell+N+\frac{N}{2}\right)\right) & p\text{ odd}\end{cases}}s_i(m-p) \end{aligned}$$

If we define
Equation:

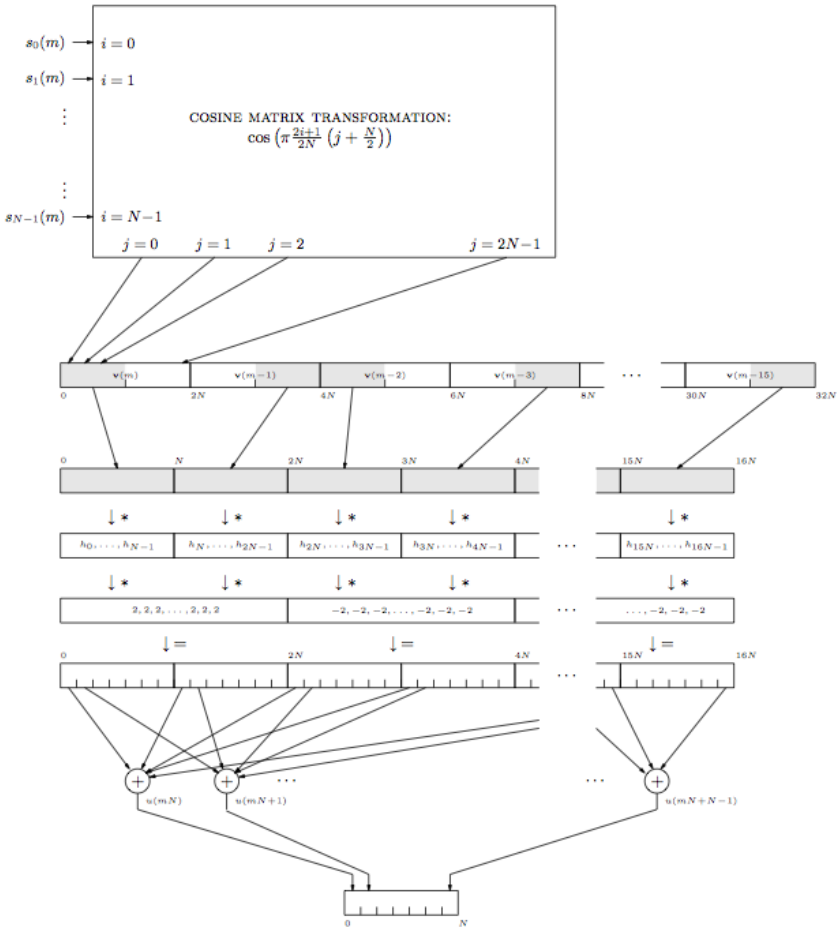
$$v_j(m) \;=\; \textstyle\sum_{i=0}^{N-1}\cos\left(\pi\frac{2i+1}{2N}\left(j+\frac{N}{2}\right)\right)s_i(m) \quad\text{for}\quad 0\le j\le 2N-1,$$

(note the range of j!) then we can rewrite
Equation:

$$u(mN+\ell) \;=\; \sum_{p=0,2,\cdots,14}(-1)^{\lfloor p/2\rfloor}h_{pN+\ell}\,v_\ell(m-p) \;+\; \sum_{p=1,3,\cdots,15}(-1)^{\lfloor p/2\rfloor}h_{pN+\ell}\,v_{\ell+N}(m-p).$$

[link] illustrates the construction of $u(mN+\ell)$ using the notation
Equation:

$$\mathbf{v}(m) \;=(v_0(m) \quad\cdots\quad v_{2N-1}(m)).$$



MPEG decoder filterbank implementation suggested in ISO/IEC 13818-3.

- *DCT Implementation of Cosine Matrixing:* As seen in [link] and [link], the filterbank implementations suggested by the MPEG standard require a cosine matrix operation that, if implemented using straightforward arithmetic, requires $32 \times 64 = 2048$ multiply/adds at both the encoder and decoder. Note, however, that the cosine transformations in [link] and [link] do bear a great deal of similarity to the DCT:
  Equation:

$$\begin{aligned} y_k &= \sqrt{\tfrac{2}{N}}\,\alpha_k \textstyle\sum_{n=0}^{N-1} x_n \cos\left(\pi \frac{2n+1}{2N} k\right); \quad k = 0 \cdots N-1, \\ &\quad \text{for} \quad \alpha_0 = 1/\sqrt{2}, \quad \alpha_{k\neq 0} = 1, \\ x_n &= \sqrt{\tfrac{2}{N}} \textstyle\sum_{k=0}^{N-1} \alpha_k\, y_k \cos\left(\pi \frac{2n+1}{2N} k\right); \quad n = 0 \cdots N-1, \end{aligned}$$

  which we know has a fast algorithm: Lee's 32×32 fast-DCT, for example, requires only 80 multiplications and 209 additions (see B.G.Lee TASSP Dec 84). So how do we implement the matrix operation using the fast-DCT? A technique has been described clearly in Konstantinides SPL 1994, the results of which are summarized below. At the encoder, the matrix operation can be written
  Equation:

$$s_i(m) = \textstyle\sum_{j=0}^{2N-1} \cos\left(\pi \frac{2i+1}{2N}\left(j - \frac{N}{2}\right)\right) w_j(m) \quad \text{for} \quad i = 0, \cdots, N-1,$$

where $\{w_0(m), \cdots, w_{2N-1}(m)$ is created from $\{x(m), \cdots, x(m-16N+1)\}$ by windowing, shifting, and adding. (See [link].) We can write

Equation:

$$s_i(m) = \sum_{j=0}^{N-1} \cos\left(\pi \frac{2i+1}{2N} j\right) \overline{w}_j(m); \quad i = 0, \cdots, N-1,$$

where, for $N = 32$, $\{\overline{w}_j(m)\}$ is the following manipulation of $\{w_j(m)\}$:

Equation:

$$\overline{w}_j(m) := \begin{cases} w_{16}(m) & j = 0 \\ w_{16+j}(m) + w_{16-j}(m) & j = 1, 2, \cdots, 16 \\ w_{16+j}(m) - w_{80-j}(m) & j = 17, 18, \cdots, 31. \end{cases}$$

Compare [link] to the inverse DCT in [link] (lower equation). At the decoder, the matrix operation can be written

Equation:

$$v_j(m) = \sum_{i=0}^{N-1} \cos\left(\pi \frac{2i+1}{2N}\left(j + \frac{N}{2}\right)\right) s_i(m) \quad \text{for} \quad j = 0, \cdots, 2N-1,$$

where $\{v_0(m), \cdots, v_{2N-1}(m)\}$ are windowed, shifted, and added to compute $\{u(m)\}$. (See [link].) It is shown in Konstantinides SPL 1994 that, for $N = 32$, $\{v_j(m)\}$ can be calculated by first computing $\{\overline{v}_j(m)\}$:

Equation:

$$\overline{v}_j(m) = \sum_{i=0}^{N-1} \cos\left(\pi \frac{2i+1}{2N} j\right) s_i(m); \quad j = 0, \cdots, N-1$$

and rearranging the outputs according to

Equation:

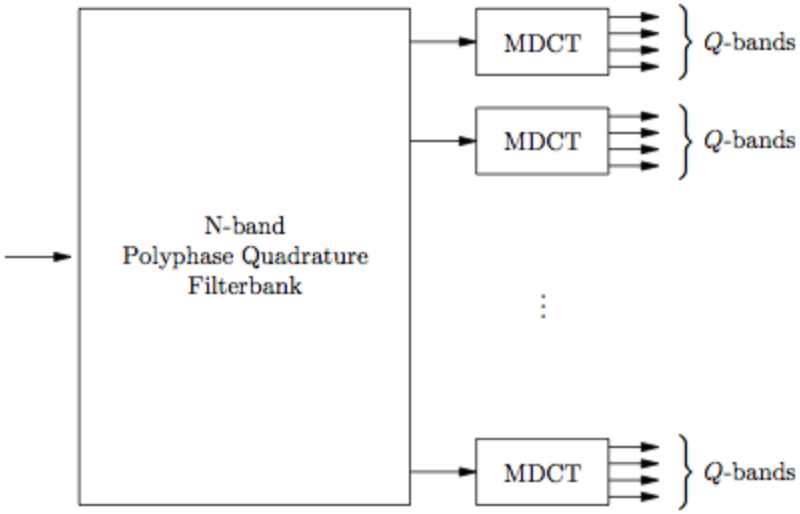
$$v_j(m) := \begin{cases} \overline{v}_{j+16}(m) & j = 0, 1, \cdots, 15, \\ 0 & j = 16, \\ -\overline{v}_{48-j}(m) & j = 17, 18, \cdots, 47, \\ -\overline{v}_{j-48}(m) & j = 48, 49, \cdots, 63. \end{cases}$$

Compare [link] to the DCT in [link] (upper equation).

MP3 and AAC: MDCT Processing
In MP3 and AAC coders, the frequency resolution of the polyphase quadrature filterbank is increased using a cascaded MDCT stage. We describe that here, and give the details of the MDCT stage.

MDCT Filterbanks

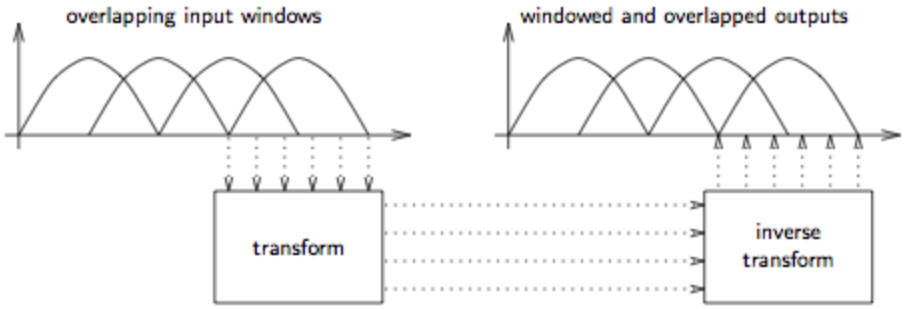
- *Hybrid Filter Banks:* In more advanced audio coders such as MPEG "Layer-3" or MPEG "Advanced Audio Coding" (the details of which will be discussed later), the 32-band polyphase quadrature filterbank (PQF) is thought to not give adequate frequency resolution, and so an additional stage of frequency division is cascaded onto the output of the PQF. This additional frequency division is accomplished using the so-called "Modified DCT" (MDCT) filterbank. (See [link].)



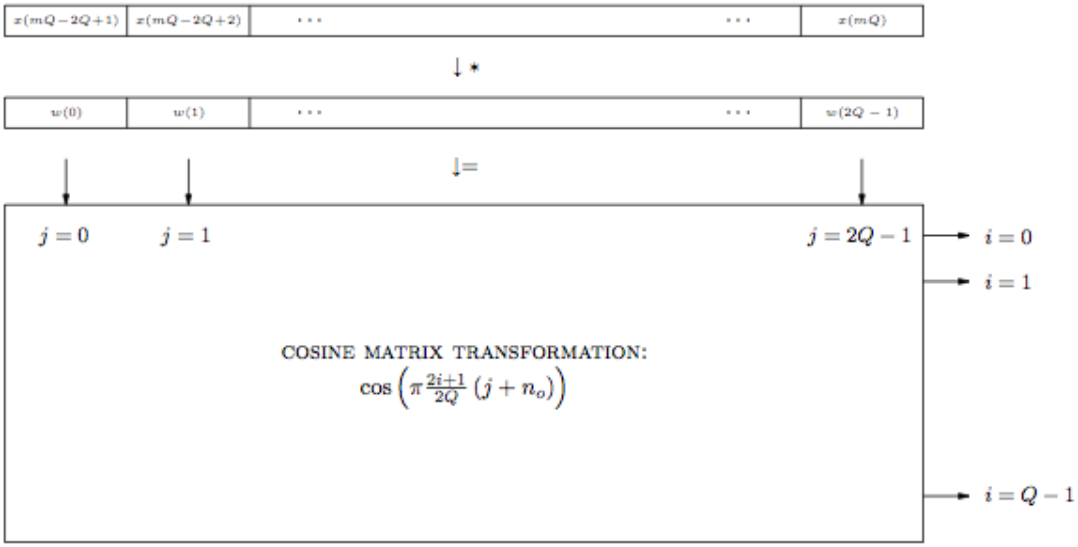
Hybrid filterbank scheme used in MPEG Layer-3 (where $N = 32$ and Q switches bewteen 6 and 18) and MPEG AAC (where $N = 4$ and Q switches between 128 and 1024).

- *Lapped Transforms:* The MDCT is a so-called "lapped transform." At the encoder, blocks of length $2Q$ which overlap by Q samples are windowed and transformed, generating Q subband samples each. At

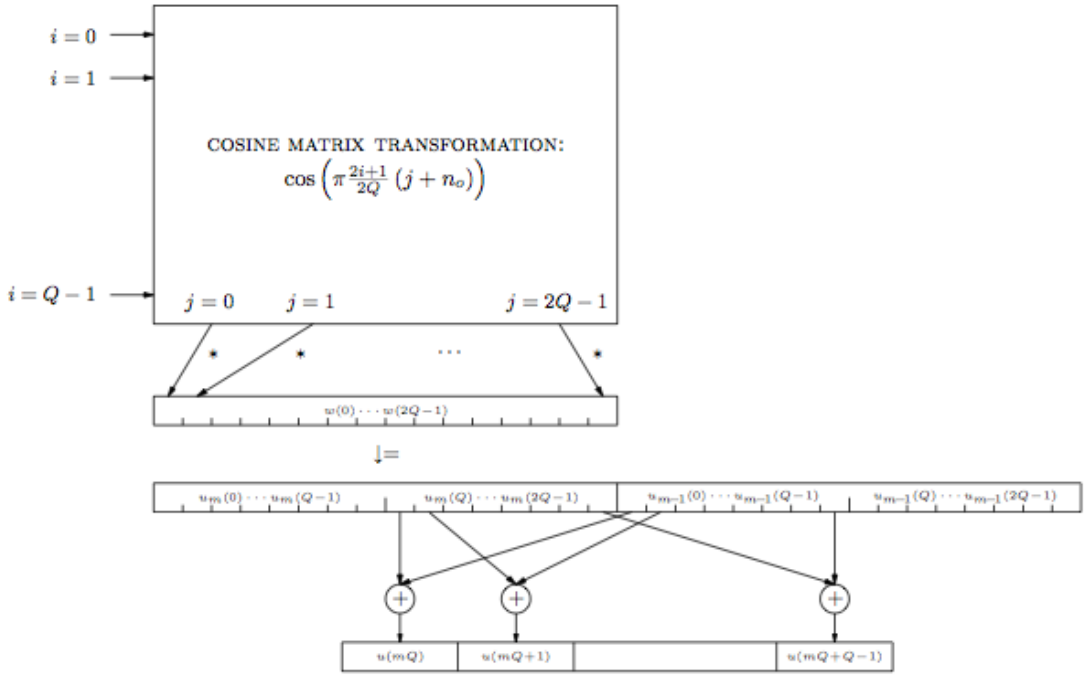
the decoder, the Q subband samples are inverse-transformed and windowed. The windowed output samples are overlapped with and added to the previous Q windowed outputs to form the output stream. [link] gives an intuitive view of the coding/decoding operation, while [link] and [link] specify the specific coder/decoder implementations used in the MPEG schemes.



A lapped transform.



MDCT filterbank: encoder implementation.



MDCT filterbank: decoder implementation.

- *Perfect Reconstruction:* Based on the cancellation of time-domain aliasing components, Princen, Johnson, & Bradley show (in ICASSP 87 and TASSP 86 papers) that the MDCT acheives perfect-reconstruction when window $\{w_n\}$ is chosen so that overlapped squared copies sum to one, i.e.,
  Equation:

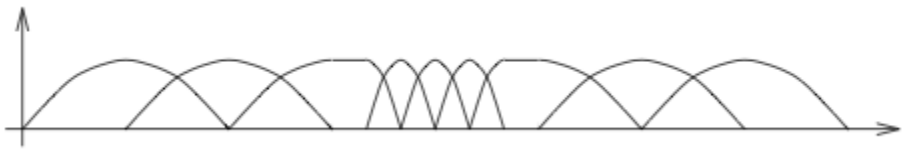
$$1 \;=\; w_{n+Q}^2 + w_n^2 \quad \text{for} \quad 0 \le n \le Q-1.$$

  The “sine” window
  Equation:

$$w_n \;= \sin\left(\frac{\pi}{2Q} n\right) \quad \text{for} \quad 0 \le n \le 2Q-1$$

  is one example of a window satisfying this requirement, and it turns out to be the one used in MPEG Layer-3.

- *Frequency Resolution:* With a window length that is only twice the number of transform outputs, we cannot expect very good frequency selectivity. But, it turns out that this is not a problem. In MPEG Layer-3, sine-window MDCTs appear at the outputs of a 32-band PQF where frequency selectivity is not a critical issue due to the limited frequency resolution of the human ear. In MPEG AAC, a 4-band PQF in conjunction with an optimized MDCT window function gives frequency selectivity just above that which current psychoacoustic models deem necessary (see M. Bosi et al., "ISO/IEC MPEG-2 Advanced Audio Coding" in JAES Oct 1997).
- *Window Switching:* Larger values of Q lead to increased frequency resolution but decreased time resolution. Time resolution is linked to the following: error due to the quantization of one MDCT output is spread out over $\approx 2QN$ time-domain output samples. For signals of a transient nature, choosing QN too high leads to audible “pre-echoes.” For less transient signals, on the other hand, the same value of QN might not be perceptible (and the increased frequency resolution might be very beneficial). Hence, most advanced coding schemes have a provision to switch between different time/frequency resolutions depending on local signal behavior. In MPEG Layer-3, for example, Q switches between 6 and 18. This is accomplished using a sine window of length 36, a sine window of length 12, and intermediate windows which are used to switch between the long and short windows while retaining the perfect reconstruction property. [link] shows an example window sequence.



Example MDCT window sequence for MPEG Layer-3.